UNIVERSITÀ DEGLI STUDI DI GENOVA

DOCTORAL THESIS

Trajectory optimization and motion planning for quadrotors in unstructured environments

---

Author:
Roberto MARINO

Supervisor:
Prof. Marco BAGLIETTO

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy

in the

Dip. di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi

Abstract

Roberto MARINO

Trajectory optimization and motion planning for quadrotors in unstructured environments

Coming out from university labs robots perform tasks usually navigating through *unstructured environment*. The realization of autonomous motion in such type of environments poses a number of challenges compared to highly controlled laboratory spaces. In unstructured environments robots cannot rely on complete knowledge of their sorroundings and they have to continously acquire information for decision making. The challenges presented are a consequence of the high-dimensionality of the state-space and of the uncertainty introduced by modeling and perception. This is even more true for aerial-robots that has a complex nonlinear dynamics a can move freely in 3D-space. To avoid this complexity a robot have to *select* a small set of relevant features, reason on a *reduced* state space and plan trajectories on *short-time horizon*. This thesis is a contribution towards the autonomous navigation of aerial robots (quadrotors) in real-world unstructured scenarios. The first three chapters present a contribution towards an implementation of Receding Time Horizon Optimal Control. The optimization problem for a model based trajectory generation in environments with obstacles is set, using an approach based on variational calculus and modeling the robots in the SE(3) Lie Group of 3D space transformations. The fourth chapter explores the problem of using minimal information and sensing to generate motion towards a goal in an indoor bulding-like scenario. The fifth chapter investigate the problem of extracting visual features from the environment to control the motion in an indoor corridor-like scenario. The last chapter deals with the problem of spatial reasoning and motion planning using atomic proposition in a multi-robot environments with obstacles.

Contents

List of Figures

List of Abbreviations

DMOC	**D**iscrete **M**echanics (and) **O**ptimal **C**ontrol
UAV	**U**nmanned **A**erial **V**ehicle
DEL	**D**iscrete **E**uler **L**agrange (equation)
DAE	**D**iscrete **A**lgebraic **E**quation
LOI	**L**ocation **O**f **I**nterest
FTS	**F**inite **T**ransition **S**ystem
APF	**A**rtificial **P**otential **F**ield
LTL	**L**inear **T**emporal **L**ogic
RRT	**R**apidly-exploring **R**andom **T**ree
NBA	**N**on-deterministic **B**uchi **A**utomata

List of Symbols

$\mathcal{L}$	Lagrangian
l	Reduced Lagrangian
q	generalized coordinates
p	generalized momentum
$\mathbb{J}$	Inertia momentum matrix
$S(\cdot), [\cdot]_x$	Skew symmetric matrix
ξ	body-fixed velocity $\in \mathfrak{se}(3)$
ω	angular velocity
Γ	Net Momentum

A Chiara

Chapter 1

Geometric Integration

1.1 Introduction

Mathematical modeling of the evolution in time of mechanical systems like robots generally includes systems of differential equations. *Solving* a physical systems means moving forward in time the model from a set of initial conditions, getting the trajectories of the involved phisical quantities. Albeit few systems can easily be solved in closed form, direct solutions of the differential equations are generally hard to solve and we need to use numerical algorithms to find a discrete temporal description of the motion. For these reason there has been a lot of research in applied mathematics, especially in the field of *numerical integrators*, leading to a various set of techniques, with different properties and performances. Moreover, these integrators are at the core of the numerical methods for trajectory optimizations. Using better integrators, in terms of speed and accuracy, received very little attention in the robotics community.

In this chapter we follow a geometric approach - instead of a traditional numerical-analytic - to the problem of integration for optimal trajectory generation of quadrotors and we will consider mechanics from a variational point of view. We start from the assumption that the essence of a mechanical systems is characterized by its *symmetries* and *invariants* (e.g. momenta) and consequently preserving this notions during the optimization algorithm is really important to capture a realistic motion. For this main reason we moves from classical integration theory to discrete variational principles, pursuing the goal of deriving robust and accurate time integrators for our optimization scheme. More precisely, we will use geometric integrator based on

the Hamilton-Pontyagrin principle, in order to deal with non-conservative external forces.

We remarks that, a classical integrator for mechanical systems advances a numerical solution along the time by adding a linear displacement in $\mathbb{R}^N$. This becomes a limitation for that systems that have a curved configuration spaces, because in this case a classical linear integration scheme does not preserve the geometrical structure of the system. A simple rigid body, for example, have an intrinsically curved configuration space that lives in the Special Euclidian Group $SE(3)$, that is the Lie group of rigid body motions (translational and rotational) obtained by the semi-direct product of $\mathbb{R}^3$ with $SO(3)$. The Special Euclidian Group, together with its Lie Algebra $\mathfrak{se}(3)$, that can be viewed as the space of the infinitesimal elements of $\mathbb{R}^3$ (i.e. the instantaneuos screw), have been used in literature to enforce that the updated poses remain within the proper group. Using the Lie Group becomes suitable because the configuration space of the systems remains a smooth curved manifold while the velocity space encoded in the associated algebra is a linear space that can be easily integrated. The mathematical setting is completed by the *group difference map* τ that maps changes in the group (smooth) in terms of changes in the algebra (linear). From a numerical point of view, Lie groups integrators exybiths not only good mathematical property but also better numerical stability and increased accuracy even with long time integration.

1.1.1 Related works

The idea of using Variational Integrator and Lie Group to build structure-preserving integrator was introduced in [28]. Conservation and numerical properties of geometric variational integrators are treated in [57]. They also proved that the properties of the used quadrature formula and underlying one-step method determine the order of accuracy of the resulting integrator. Leok and Shingel illustrate in [50] systematic methods for constructing numerical integrators. An application for the full-body problem is presented by Lee at al in [48] while the dynamics of the 3d pendulum system is the topic of [49]. Structure preserving properties of variational integrators based on Hamilton-Pontryagin principle are treated in [10]. A comprehensive introduction on Lie group theory is in [1], a treatment of Lie Groups 2D and

3D transformations can be found in [18] while interpolation of trajectory generated by rigid motion in 3D is introduced in [17].

1.2 Variational Calculus and Dynamical Systems

Considering mechanics from a variational point of view lead us to the work of Euler, Lagrange and Hamilton. The corner-stone theorem is set by Hamilton and is known as *Hamilton's Principle* or the *Least Action Principle*: it says that *every mechanical system follow an optimal trajectory from its starting position to its final position*. The main contribution of this principle is that we can turn our way of thinking about classical mechanics: the trajectory of a body subjected to an applied force has optimal geometric properties, like a geodesic on a curved surface. Notice that the relevance of this point of view is also valid for elctromagnetism and quantum mechanics.

Let be q the state variable that parametrize a finite-dimensional dynamical system. The Lagrangian function of the system is given as a function of q and $\dot{q}$. In the more restrictive case of basic elasticity, this Lagrangian function $\mathcal{L}$ is defined as the kinetic energy K minus the potential energy W ot the system:

$$\mathcal{L}(q, \dot{q}) = K(\dot{q}) - W(q) \tag{1.1}$$

The integral of $\mathcal{L}$ alongh a path $q(t)$ over time $t \in [0, T]$ is the *action* function of the system. Hamilton's principle now states that the correct path of motion of a dynamical system is such that its action has a stationary value, i.e. the integral along the correct path has the same value to within first-order infinitesimal perturbations. The entire motion of a systems between two fixed times, $T = t_2 - t_1$ is subjected to this "integral principle":

$$S[q, t1, t2] = \int_0^T \mathcal{L}[q(t), \dot{q}(t), t]dt \tag{1.2}$$

and the action principle states that:

$$\delta S[q, t1, t2] = 0 \tag{1.3}$$

To account for non-conservative forces F, the least action principle can be modified:

$$\delta \int_0^T \mathcal{L}(q(t), \dot{q}(t))dt + \int_0^T F(q(t), \dot{q}(t))\delta q dt = 0 \tag{1.4}$$

which is known as *Lagrange-d'Alembert principle*.

1.2.1 Hamilton-Pontryagin Principle

The Hamilton-Pontryagin principle, that is a generalization of the Hamilton's principle, states that the equations of mechanics are given by the critical points of the *Hamilton-Pontryagin action integral*:

$$\delta \int_0^T [p(t)(\dot{q}(t) - v(t)) + \mathcal{L}(q(t), v(t))]dt = 0 \tag{1.5}$$

where the configuration variable q, the velocity v and the momentum p are all viewed as indipendent variables. The similarity is evident w.r.t Hamilton's principles: indeed p can be viewed as Lagrange multiplier that impose the equality between $\dot{q}$ and v. The Hamilton-Pontryagin principle yields equations equivalent to the Euler-Lagrange equations, since, for the variations $\delta p(t)$, $\delta q(t)$ and $\delta v(t)$ over the three independent variables, we get:

$$v = \dot{q} \tag{1.6}$$

$$\frac{dp}{dt} = \frac{\partial \mathcal{L}(q, v)}{\partial q} \tag{1.7}$$

$$p = \frac{\partial \mathcal{L}(q, v)}{\partial v} \tag{1.8}$$

Thanks to the fact that the formulation involves both phase space variables q and v within the action integral, these equations can be framed from a Lagrangian and Hamiltonian point of view.

1.3 Geometric Integration for long time horizon evolutions

As we already stated there are a lot of problems in science and engineering, modeled by Ordinary Differential Equation, that has to be simulated within *a long time horizon*: their fields span from molecular dynamics to astronomy. This problems are generally nonlinear and sensitive to small perturbations of the initial parameters. Simulating this problems in an euclidian configuration spaces is often expensive and imprecise. *Geometric Integrators* are numerical methods that exploit the geometric structure of the underlying dynamical systems from a qualitative point of view deriving the integration scheme from the discrete variational principles. The theory includes discrete analogs of the Lagrangian, Noether's theorem, Euler Lagrange equations and Legendre transformation.

The idea behind variational integrators is to discretize the action with respect to time before finding the discrete time equations of motions. Doing so leads to integration schemes that avoid common problems associated with numerically integrating a continuous differential equation. These problems can occur because the numerical approximations that are introduced do not respect fundamental mechanical properties like conservation of momentum, energy, and symplectic form, all of which are relevant to mechanical systems. The dynamics of a mechanical system in continuous-time is described by the Euler-Lagrange equation where q is the vector of generalized coordinates, u is the vector of control inputs, $\mathcal{L} = K - W$ is the Lagrangian, with K is the kinetic energy and W is the potential energy.

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = F(q, \dot{q}, u) \tag{1.9}$$

The Euler-Lagrange equations can be derived from extremizing the action integral, typically referred to as the least action principles. The action integral, i.e. the integral of the Lagrangian with respect to time along an arbitrary curve in the tangent bundle, says that a mechanical systems will follow the trajectory that extremizes the action with respect to variation in $q(t)$. Applying calculus of variations to the action integral shows that it is extremized by the Euler-Lagrange equations. A variational integrator is derived by choosing a discrete Lagrangian, $\mathcal{L}_d$ that approximates the

action over a discrete time step.

$$\mathcal{L}_d(q_k, q_{k+1}) \approx \int_{t_k}^{t_{k+1}} \mathcal{L}(q(s), \dot{q}(s))ds \tag{1.10}$$

where q_k is a discrete-time configuration that appoximates the trajectory and can be achieved using any quadrature rule.

By summing the discrete Lagrangian over an arbitrary trajectory, the action integral is approximated by a discrete action. The action principle is then applied to the action sum to find the discrete trajectory that extremizes the discrete action. The result is the Discrete Euler-Lagrange equation (DEL):

$$D_2\mathcal{L}_d(q_{k-1}, q_k) + D_1\mathcal{L}_d(q_k, q_{k+1}) = 0 \tag{1.11}$$

where $D_n\mathcal{L}_d$ is the derivative of $\mathcal{L}_d$ w.r.t. the n-th argument.

The DEL equations depend on the previous, current and future configuration, and they do not depend on the velocity, making this schema appealing for embedded systems that measure configurations but not velocities. The DEL equation can also be written in an equivalent $position - momentum$ form that only depends on the current and future time steps

$$p_k + D_1\mathcal{L}_d(q_k, q_{k+1}) = 0 \tag{1.12}$$

$$p_{k+1} = D_2\mathcal{L}_d(q_k, q_{k+1}) \tag{1.13}$$

where p_k is the discrete generalize momentum of the system at time k. By these definitions, it should be clear that $-D_1\mathcal{L}_d(q_k, q_{k+1})$ and $D_2\mathcal{L}_d(q_k, q_{k+1})$ are both playing the role of a Legendre transform in discrete time, and are accordingly referred to as the left and right Legendre transforms, respectively.

Equations (1.12-13) impose a constraint on the current and future position and momenta. Given an initial state p_k and q_k the implicit non-linear equation (1.12) is solved numerically (not explicitly) to find the next configuration q_{k+1} using a numeric method such as the Newton-Raphson algorithm. The next momentum then is computed explicitly using (1.13). After an update, k is incremented and the process

is repeated to simulate the system for as many steps as desired.

Variational integrators can be extended to include non-conservative forcing by using a discrete form of the Lagrange-d'Alembert principle. The continuous force is approximated by a left and right discrete force F_d^- and F_d^+:

$$\int_{t_k}^{t_{k+1}} F(q(s), \dot{q}(s), u(s))\delta q ds = F_d^-(q_k, q_{k+1}, u_k)\delta q_k + F_d^+(q_k, q_{k+1}, u_k)\delta q_{k+1} \quad (1.14)$$

where u_k is the discretization of the continuous force inputs: $u_k = u(t_k)$ As with the discrete Lagrangian, the discrete forcing can be approximated by any quadrature rule. For clarity, we use the following abbreviations for the discrete Lagrangian and discrete forces throughout this thesis:

$$\mathcal{L}_k = \mathcal{L}_d(q_{k-1}, q_k) \quad (1.15)$$

$$F_k^{\pm} = F_d^{\pm}(q_{k-1}, q_k, uk - 1) \quad (1.16)$$

The forced DEL equations are then given by:

$$p_k + D_1\mathcal{L}_{k+1} + F_{k+1}^- = 0 \quad (1.17)$$

$$p_{k+1} = D_2\mathcal{L}_{k+1} + F_{k+1}^+ \quad (1.18)$$

1.3.1 Discrete Hamilton-Pontryagin formulation

Here we present the discretization of principles introduced in section 1.3.1. A motion q(t) for $t \in [0, T]$ is replaced by a discrete sequence of poses q_k with $k = 0, \ldots, N$. We introduce h_k as the time step between time t_k, t_{k+1}. Note that the time step can be adjusted throughout the computation based on standard time step control ideas if necessary. We similarly discretize $v(t)$ an $p(t)$ by the sets v_k and p_k, respectively velocity and momenta.

For a given choice of Lagrangian, one can easily derive a discrete action through quadrature:

$$\mathcal{L}_d(q_k, v_{k+1}) = \mathcal{L}(q_k + \alpha h_k v_{k+1}, v_{k+1}) \approx \int_{t_k}^{t_{k+1}} \mathcal{L}(q, \dot{q})dt \quad (1.19)$$

Once a discrete Lagrangian is given a discrete Hamilton-Pntryagin principle can be expressed:

$$\delta \sum_{k=0}^{N} \left[p_{k+1} \left(\frac{q_{k+1} - q_k}{h_k} - v_{k+1} \right) h_k + \mathcal{L}_d(q_k, v_{k+1}) \right] = 0 \tag{1.20}$$

The discrete Hamitlon-Pontryagin principle yields, upon taking discrete variations with respect to each state variable with fixed endpoints, to the following equations:

$$\boldsymbol{\delta p} : q_{k+1} - q_k = h_k v_{k+1} \tag{1.21}$$

$$\boldsymbol{\delta q} : p_{k+1} - p_k = D_1 \mathcal{L}_d(q_k, v_{k+1}) \tag{1.22}$$

$$\boldsymbol{\delta v} : h_k p_{k+1} = D_2 \mathcal{L}_d(q_k, v_{k+1}) \tag{1.23}$$

where D_1 and D_2 denote the differentiation with respect to the first q_k and second v_{k+1} arguments of $\mathcal{L}_d$. The resulting DEL equation in this case is:

$$D_2 \mathcal{L}_d(q_k, v_{k+1}) - h_k p_k - h_k D_1 \mathcal{L}_d(q_k, v_{k+1}) = 0 \tag{1.24}$$

can now be solved for v_{k+1} with any non-linear solver, and q_{k+1}, p_{k+1} are found using eqs (1.21) and (1.22).

1.4 Discrete Pontryagin-d'Alembert Principle

For non-conservative systems, the continuous Pontryagin-d'Alembert principle is given by:

$$\delta \int_0^T [\mathcal{L}(q, v) + p(\dot{q} - v)] dt + \int_0^T F_v(q, v) \delta q dt = 0 \tag{1.25}$$

where $F(q, v)$ is an arbitrary non-conservative force function. The discrete Pontryagin-d'Alembert principle can thus be defined as

$$\begin{aligned} \delta \left(\sum_{k=0}^{N} p_{k+1}(q_{k+1} - q_k - h_k v_{k+1}) + \mathcal{L}_d(q_k, v_{k+1}) \right) + \\ \sum_{k=0}^{N} \left(F_d^-(q_k, v_{k+1}) \delta q_k + F_d^+(q_k, v_{k+1}) \delta q_{k+1} \right) = 0 \end{aligned} \tag{1.26}$$

where $F_d^{\mp}$ approximate the total forcing over a time step through:

$$F_d^-(q_k, v_{k+1})\delta q_k + F_d^+(q_k, v_{k+1})\delta q_{k+1} \approx \int_{t_k}^{t_{k+1}} F(q, \dot{q})\delta q dt \tag{1.27}$$

This yields, upon taking discrete variations, the following forced discrete variational equations:

$$\boldsymbol{\delta p} : q_{k+1} - q_k = h_k v_{k+1} \tag{1.28}$$

$$\boldsymbol{\delta q} : p_{k+1} - p_k = D_1\mathcal{L}_d(q_k, v_{k+1})) + F_d^-(q_k, v_{k+1}) + F_d^+(q_{k-1}, v_k) \tag{1.29}$$

$$\boldsymbol{\delta v} : k_k p_{k+1} = D_2\mathcal{L}_d(q_k, v_{k+1})) \tag{1.30}$$

Our integration scheme can also accomodate holonomic constraints, i.e., contraints described by $g(q) = 0$. One just need to write the Hamilton-Pontryagin principle in terms of the variable q while using Lagrange multipliers λ to impose $g(q) = 0$:

$$\delta \int_0^T [\mathcal{L}(q, v) + p(\dot{q} - v)]dt + \lambda g(q) = 0 \tag{1.31}$$

The discrete counterpart is then given by:

$$\delta \sum_{k=0}^{N} \left[p_{k+1}\left(q_{k+1} - q_k - h_k v_{k+1}\right) h_k + \mathcal{L}_d(q_k, v_{k+1}) + h_k \lambda_{k+1}\, g(q_{k+1})\right] = 0 \tag{1.32}$$

which yelds to the following contrained discrete Hamilton-Pontryagin equations:

$$\boldsymbol{\delta p} : q_{k+1} - q_k = h_k v_{k+1} \tag{1.33}$$

$$\boldsymbol{\delta q} : p_{k+1} - p_k = D_1\mathcal{L}_d(q_k, v_{k+1})) + h_k \lambda_k \nabla g(q_k) \tag{1.34}$$

$$\boldsymbol{\delta v} : k_k p_{k+1} = D_2\mathcal{L}_d(q_k, v_{k+1})) \tag{1.35}$$

$$g(q_{k+1}) = 0 \tag{1.36}$$

1.5 Leaving the flatland: Lie Algebras

Lie algebras were introduced to study the concept of *infinitesimal transformations* by Marius Sophus Lie in the 1870s and independently discovered by Wilhelm Killing

in the 1880s. The Lie Algebra $\mathfrak{g}$ associated to each Lie group $\mathfrak{G}$ is the *tangent space* built around the identity element of the group and has a basis composed by elements called *generators*. All tangent vectors are a linear combination of the generators. The tangent space associated with a Lie group provides a suitable space to represents differential quantities (e.g. velocities). Every Lie Algebra is accompained by operators that maps elements from the algebra to the group and vice-versa:

- the *exponential map* $exp : \mathfrak{g} \rightarrow \mathfrak{G}$ also called τ map
- the *logaritmic map* $log : \mathfrak{G} \rightarrow \mathfrak{g}$

1.5.1 SO(3): Lie Group of rotations in 3D space

Rotations in 3D space are represented by the elements of the $SO(3)$ group. Composition and inversion in the group correspond to matrix multiplication and inversion. Because rotation matrices are orthogonal, inversion is equivalent to transposition:

$$\boldsymbol{R} \in SO(3) \tag{1.37}$$

$$\boldsymbol{R}^{-1} = \boldsymbol{R}^{T} \tag{1.38}$$

The Lie algebra, $\mathfrak{so}(3)$ is the set of $3x3$ skew-symmetric matrices. The generators are the derivatives of rotation around the standard axis, evaluated at the identity.

$$B_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$B_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

$$B_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{\omega} \in \mathbb{R}^3 \tag{1.39}$$

$$\boldsymbol{\omega}_x = \boldsymbol{\omega}_1 B_1 + \boldsymbol{\omega}_2 B_2 + \boldsymbol{\omega}_3 B_3 \in \mathfrak{so}(3) \tag{1.40}$$

$$S(\boldsymbol{\omega}) = \boldsymbol{\omega}_x = \begin{bmatrix} 0 & -\boldsymbol{\omega}_3 & \boldsymbol{\omega}_2 \\ \boldsymbol{\omega}_3 & 0 & -\boldsymbol{\omega}_1 \\ -\boldsymbol{\omega}_2 & \boldsymbol{\omega}_1 & 0 \end{bmatrix} \tag{1.41}$$

The exponential map exp for $SO(3)$ is the Rodrigues formula:

$$exp(\boldsymbol{\omega}_x) = \boldsymbol{I} + \left(\frac{\sin\theta}{\theta}\right)\boldsymbol{\omega}_{\boldsymbol{x}} + \left(\frac{1-\cos\theta}{\theta^2}\right)\boldsymbol{\omega}_x^2 \tag{1.42}$$

where

$$\theta^2 = \boldsymbol{\omega}^T\boldsymbol{\omega} \tag{1.43}$$

The exponential map can be inverted to give the *log* map, going from SO(3) to $\mathfrak{so}(3)$:

$$\boldsymbol{R} \in SO(3) \tag{1.44}$$

$$\theta = \arccos\left(\frac{tr(\boldsymbol{R})-1}{2}\right) \tag{1.45}$$

$$log(\boldsymbol{R}) = \frac{\theta}{2\sin\theta}(\boldsymbol{R} - \boldsymbol{R}^T) \tag{1.46}$$

1.5.2 SE(3): Lie Group of Rigid transformations in 3D space

The group of rigid transformation in 3D space, $SE(3)$, is well represented by linear transformations on homogeneous four-vectors:

$$\boldsymbol{R} \in SO(3), \boldsymbol{t} \in \mathbb{R}^3 \tag{1.47}$$

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix} \in SE(3) \tag{1.48}$$

The Lie algebra $\mathfrak{se}(3)$ is the set of 4x4 matrices corresponding to differential translations and rotations. The six generators are:

$$B_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B_5 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B_6 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

An element of the algebra is the linear combination of the generators:

$$(\boldsymbol{v}, \boldsymbol{\omega}) \in \mathbb{R}^6 \tag{1.49}$$

$$(\boldsymbol{v}, \boldsymbol{\omega}) = \boldsymbol{v}_1 B_1 + \boldsymbol{v}_2 B_2 + \boldsymbol{v}_3 B_3 + \boldsymbol{\omega}_4 B_4 + \boldsymbol{\omega}_5 B_5 + \boldsymbol{\omega}_6 B_6 \in \mathfrak{se}(3) \tag{1.50}$$

The exponential map has a closed form representation given by:

$$\boldsymbol{v}, \boldsymbol{\omega} \in \mathbb{R}^3 \tag{1.51}$$

$$\theta = \sqrt{\boldsymbol{\omega}^T \boldsymbol{\omega}} \tag{1.52}$$

$$A = \frac{sin\theta}{\theta} \tag{1.53}$$

$$B = \frac{1 - cos\theta}{\theta} \tag{1.54}$$

$$C = \frac{1 - A}{\theta^2} \tag{1.55}$$

$$\boldsymbol{R} = \boldsymbol{I} + A\boldsymbol{\omega}_x + B\boldsymbol{\omega}_x^2 \tag{1.56}$$

$$\boldsymbol{V} = \boldsymbol{I} + B\boldsymbol{\omega}_x + C\boldsymbol{\omega}_x^2 \tag{1.57}$$

$$exp(\boldsymbol{v}, \boldsymbol{\omega}) = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{V}\boldsymbol{v} \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{1.58}$$

The matrix $\boldsymbol{V}$ has a closed form inverse:

$$\boldsymbol{V}^{-1} = \boldsymbol{I} - \frac{1}{2}\boldsymbol{\omega}_x + \frac{1}{\theta^2}\left(1 - \frac{A}{2B}\right)\boldsymbol{\omega}_x^2 \tag{1.59}$$

The logarithm map for $SE(3)$ computes the $\boldsymbol{R}$ matrix as in the $SO(3)$ case, then compute the $\boldsymbol{t}$ vector as $\boldsymbol{v} = \boldsymbol{V}^{-1}\boldsymbol{t}$

1.6 Integration of dynamical systems in the Euclidian Space

A controlled dynamic system in continuous time is usually modeled by an ordinary differential equation (ODE) in the Euclidian Space $\mathbb{R}^N$ on a horizon $[0, T]$ with controls $u(t)$ by

$$\dot{x}(t) = f(x(t), u(t), t), \quad t \in [0, T] \tag{1.60}$$

where t is the continuous time and $x(t)$ is the state vector. We define *Initial Value Problem* (IVP) the problem given by eq (1.60) along with the constraint $x(0) = x_0$. The uniqueness and existence of a solution for the IVP is guaranteed under continuity by Peano and under finitely many discontinuities by Picard and Lindelof. Numerical integration methods are used to approximately solve an IVP. They come in different variants and can be categorize along two major categories:

- one-step methods
- multi-step methods

or

- explicits methods
- implicits methods

A numerical integration method start discretizing the time t with resolution $\Delta t = T/N$ with N the number of timesteps and $t_k := k\Delta t$ and the solution is approximated on the grid points by values s_k that shall satisfy $s_k \approx x(t_k)$ for $k = 0, \ldots, N-1$. Numerical integration methods differ in the ways how they approximate the solution on the grid points and between, but they shall have the propery that if $N \to \infty$ then $s_k \to x(t_k)$. This is called *convergence*. The simplest integrator is the explicit Euler method. It first sets $s_0 := x_0$ and then proceeed using the rules

$$s_{k+1} := \Delta t \; f(s_k, t_k) \tag{1.61}$$

It is a low order method and due to this low order it is very inefficient and should not be used in practice. However, with a few extra evaluations of f in each step, higher order one-step methods can easily be obtained, the *explicit Runge-Kutta (RK) methods*. This methods evaluates m times the dynamic function at each step at the intermediate states $s_k^{(i)}$ with $i = 1, \ldots, m$, that live on a grid of intermediate time points $t_k^{(i)} := t_k + c_i \Delta t$, with c_i suitably chosen and $c_i \in [0, 1]$. The one step RK method is characterized by the equation:

$$s_{k+1} := s_k + \Delta t \sum_{j=1}^{m} b_j f(s_k^{(j)}, t_k^{(j)}) \tag{1.62}$$

When an explicit integrator is applied to a very stable system, its steps very easily overshoot. For example, when we want to simulate using the explicit Euler method for a very large $\lambda \gg 1$ the ODE

$$\dot{x} = -\lambda x \tag{1.63}$$

that is superstable and converges very quickly to zero, we observe that the simulated system becomes unstable for $\Delta t > \frac{2}{\lambda}$. In this case we can use the implicit Euler integrator, which in each integrator step solves the nonlinear equation in the variable s_{k+1}

$$s_{k+1} = s_k + \Delta t f(s_{k+1}, t_{k+1}) \tag{1.64}$$

1.7 Lie Group Variational Integrators

The idea at the base of the Lie Group Variational Integrator (LGVI) is to express the update map in terms of the exponential map

$$g_1 = g_0 \exp(\xi_{01}) \tag{1.65}$$

where $\xi_{01} \in \mathfrak{g}$. Denoting the Lie Group configuration of our systems as X, with $x \in X$ the dynamics can be written as

$$\dot{x} = x\, \hat{f}(t, x, u) \tag{1.66}$$

which is a generalization of equation (1.65). The Lie Algebra element $f(t, x, u) \in \mathbb{R}^N \approx T_e X$ is interpreted as the body-fixed state velocity and the product $x\hat{f}$ is the tangent group action of x. A time-update $x_k \rightarrow x_{k+1}$ is performed by evolving a geodesic motion on the group, leading to:

$$x_{k+1} = x_k \exp(f_k) \tag{1.67}$$

where f_k is a discrete approximation of a continuous flow f. The exp function is the standard choice for the *group difference map* τ, i.e. a map that expresses changes in the group in terms of elements in its Lie algebra. Another choice for τ, valid only for certain matrix groups (SO(3), SE(2) and SE(3)) is the *Cayley Map*

$$cay : \mathfrak{g} \rightarrow \mathfrak{G}, cay(\xi) = \left(e - \frac{\xi}{2}\right)^{-1} \cdot \left(e + \frac{\xi}{2}\right) \tag{1.68}$$

Although this last map provide only an approximation to the integral curve defined by exp, we include it because it is very easy to compute and thus results in a more efficient implementation.

1.8 Application to the Full Body Problem

The full body problem studies the dynamics of a set of rigid bodies $\mathbb{B}_i$ subjected to a potential field, where the potential depends on the position and attitue of the body.

Therefore the translational and the rotational dynamics are coupled. First of all, we will introduce the continuous setting of the problem and then the discrete procedure to obtain a Lie group variational integrator of the problem. Let be $I = (e1, e2, e3)$ an inertial frame and $B = (e_b^1, e_b^2, e_b^3)$ a body fixed frame attached to the i-th body $\mathbb{B}_i$. The configuration space of the i-th rigid body is $SE(3)$ and x_i, R_i are respectively the position and orientation of the body in the inertial frame, and ω_i the angular velocity. Let be $\mathbb{J}$ the principal inertia momentum matrix and m_i the total mass ot the i-th body.

1.8.1 Building the Lagrangian

To derive the equations of motion we first build the Lagrangian for the dynamic problem. Given $(x_i, R_i) \in SE(3)$ the inertial position of a mass element is given by $x_i + R_i\rho_i$, where $\rho_i \in \mathbb{R}^3$ denotes the position of the mass element in the body fixed frame B. Then the kinetic energy of the mass element can be written as

$$T_i = \frac{1}{2}\int_{\mathbb{B}_i} \left\|\dot{x}_i + \dot{R}_i\rho_i\right\|^2 dm_i \tag{1.69}$$

Using the fact that $\int_{\mathbb{B}_i} \rho_i dm_i = 0$ and the kinematic equation $\dot{R}_i = R_i S(\Omega_i)$, the kinetic energy T_i can be rewritten as

$$T_i(\dot{x}_i, \omega_i) = \frac{1}{2}\int_{\mathbb{B}_i} \|\dot{x}_i\|^2 + \|S(\omega_i)\rho_i\|^2 dm_i = \frac{1}{2}m_i\|\dot{x}\|^2 + \frac{1}{2}tr[S(\omega_i)\,\mathbb{J}_i\,S(\omega_i)^T] \tag{1.70}$$

where m_i is the total mass of $\mathbb{B}_i$ and J_i is the moment of inertia matrix.

Then the Lagrangian for n full bodies can be written as

$$\mathcal{L}(x, \dot{x}, R, \omega) = \sum_{i=1}^{n} \frac{1}{2}m_i\|\dot{x}_i\|^2 + \frac{1}{2}tr[S(\omega_i)\,\mathbb{J}_i\,S(\omega_i)^T] - U(x, R) \tag{1.71}$$

where U(x,R) is the potential energy.

1.8.2 Compute the action integral

The action integral is defined to be:

$$\mathbb{A} = \int_{t_0}^{t_f} \mathcal{L}(x, \dot{x}, R, \omega) dt \tag{1.72}$$

and its variation is

$$\delta\mathbb{A} = \sum_{i=1}^{n} \int_{t_0}^{t_f} m_i \dot{x}_i^T \delta\dot{x}_i - \frac{\partial U^T}{\partial x_i} \delta x_I + \tag{1.73}$$
$$\frac{1}{2} tr[-\dot{\eta}_i S(\mathbb{J}_i \omega_i)] + \frac{1}{2} tr\left[\eta_i \left[S(\omega_i \times \mathbb{J}_i \omega_i) + 2R_i^T \frac{\partial U}{\partial R_i}\right]\right] dt$$

In summary, the continuous equations of motion for the full body problem, in Lagrangian form can be written for $i \in (1, 2, \ldots, n)$ as:

$$\dot{v}_i = -\frac{1}{m_i} \frac{\partial U}{\partial x_i} \tag{1.74}$$

$$\mathbb{J}_i \dot{\omega}_i + \omega_i \times \mathbb{J}_i \omega_i = \Gamma_i \tag{1.75}$$

$$\dot{x}_i = v_i \tag{1.76}$$

$$\dot{R}_i = R_i S(\omega_i) \tag{1.77}$$

and in Hamiltonian form:

$$\dot{\gamma}_i = -\frac{\partial U}{\partial x_i} \tag{1.78}$$

$$\mathbb{J}_i \dot{\Pi}_i + \omega_i \times \Pi_i = \Gamma_i \tag{1.79}$$

$$\dot{x}_i = \frac{\gamma_i}{m_i} \tag{1.80}$$

$$\dot{R}_i = R_i S(\omega_i) \tag{1.81}$$

1.8.3 Discrete case

In continuous time, the structure of the kinematics ensure that R_i evolve automatically in SO(3). Here we have to introduce a new variable $F_{i_k} \in SO(3)$ defined such

that

$$R_{i_{k+1}} = R_{i_k} F_{i_k} \tag{1.82}$$

$$F_{i_k} = R_{i_k}^T R_{i_{k+1}} \tag{1.83}$$

The skew symmetric matrix can be approximated as

$$S(\omega_{i_k}) = R_{i_k}^T R_{i_k} \approx R_{i_k}^T \frac{R_{i_{k+1}} - R_{i_k}}{h} = \frac{1}{h}(F_{i_k} - I_{3\times 3}) \tag{1.84}$$

The velocity $\dot{x}_{i_k}$ can be approximated simply by $(x_{i_{k+1}} - x_{i_k})/h$. Using these approximations of the angular and linear velocity, the kinetic energy of the i-th body can be approximated as:

$$\begin{aligned} T_i(\dot{x}_i, \omega_i) &\approx T_i\left(\frac{1}{h}(x_{i_{k+1}} - x_{i_k}), \frac{1}{h}(F_{i_k} - I_{3\times 3})\right) \\ &= \frac{1}{2h^2} m_i \left\| x_{i_{k+1}} - x_{i_k} \right\|^2 + \frac{1}{h^2} tr[(I_{3\times 3} - F_{i_k})\mathbb{J}_i] \end{aligned} \tag{1.85}$$

A discrete Lagrangian is constructed such that it approximates a segment of the action integral:

$$\mathcal{L}_d = \frac{h}{2}\mathcal{L}\left(x_k, \frac{1}{h}(x_{k+1} - x_k), R_k, \frac{1}{h}(F_k - I)\right) \tag{1.86}$$

$$\begin{aligned} &+ \frac{h}{2}\mathcal{L}\left(x_{k+1}, \frac{1}{h}(x_{k+1} - x_k), R_{k+1}, \frac{1}{h}(F_k - I)\right) \\ &= \sum_{i=1} n \frac{1}{2h} m_i \left\| x_{i_{k+1}} - x_{i_k} \right\|^2 + \frac{1}{h} tr[(I_{3\times 3} - F_{i_k})\mathbb{J}_i] \end{aligned} \tag{1.87}$$

$$- \frac{h}{2}U(x_k, R_k) - \frac{h}{2}U(x_{k+1}, R_{k+1})$$

The discrete action is defined as:

$$\mathbb{A}_d = \sum_{k=0}^{N-1} \mathcal{L}_d(x_k, x_{k+1}, R_k, F_k) \tag{1.88}$$

Computing the variation of the action and applying the Hamilton principle we obtain the discrete equations of motion for the full body problem in Lagrangian form:

$$\frac{1}{h}(x_{i_{k+1}} - 2x_{i_k} - x_{i_{k-1}}) = -h\frac{\partial U_k}{\partial x_{i_k}} \tag{1.89}$$

$$\frac{1}{h}(F_{i_{k+1}}\mathbb{J}_i - \mathbb{J}_i F_{i_{k+1}}^T - \mathbb{J}_i F_{i_k} + F_{i_k}^T \mathbb{J}_i) = hS(\Gamma_{i_{k+1}}) \tag{1.90}$$

$$R_{i_{k+1}} = R_{i_k} F_{i_k} \tag{1.91}$$

The discrete equations of motion for the full body problem, in Hamiltonian form is:

$$x_{i_{k+1}} = x_{i_k} + \frac{h}{m_i}\gamma_{i_k} - \frac{h^2}{2m_i}\frac{\partial U_k}{\partial x_{i_k}} \tag{1.92}$$

$$\gamma_{i_{k+1}} = \gamma_{i_k} - \frac{h}{2}\frac{\partial U_k}{\partial x_{i_k}} - \frac{h}{2}\frac{\partial U_{k+1}}{\partial x_{i_{k+1}}} \tag{1.93}$$

$$hS\left(\Pi_{i_k} + \frac{h}{2}\Gamma_{i_k}\right) = F_{i_k}\mathbb{J}_i - \mathbb{J}_i F_{i_k}^T \tag{1.94}$$

$$\Pi_{i_{k+1}} = F_{i_k}^T \Pi_{i_k} + \frac{h}{2}F_{i_k}^T \Gamma_{i_k} + + \frac{h}{2}\Gamma_{i_{k+1}} \tag{1.95}$$

$$R_{i_{k+1}} = R_{i_k} F_{i_k} \tag{1.96}$$

Chapter 2

Trajectory Optimization

A trajectory optimization or optimal control problem can be formulated as a collection of N *phases*. In general, the independent variable t for phase k is defined in the region $t_0^k \leq t \leq t_f^k$. For many applications the independent variable t is the time, and the phases are sequential. Whitin phase k the dynamics of the system are described by a set of *dynamic* variables

$$z = \begin{bmatrix} x^k(t) \\ u^k(t) \end{bmatrix}$$

made up of the n_x^k state variables and n_u^k control variables respectively. In addition, the dynamics may incorporate the n_p^k parameters p^k which are not dependent on t. For clarity we drop the phase-dependent notation from the remaining discussion in this chapter, however it is important to remember that many complex problem descriptions require different dynamics and/or constraints within each phase, and the formulation accomodates this requirement.

Typically the dynamics of the system are defined by a set of ordinary differential equations written in explicit form, which are referred to as the *state* or *system equations*

$$\dot{x} = f\left[x(t), u(t), p, t\right] \tag{2.1}$$

where x is the n_x dimension state vector. Initial conditions at time t_0 are defined by

$$\psi_{0l} \leq \psi\left[x(t_0), u(t_0), p, t_0\right] \leq \psi_{0u} \tag{2.2}$$

where $\psi\left[x(t_0), u(t_0), p, t_0\right] \equiv \psi_0$ and terminal conditions at the final time t_f are defined by

$$\psi_{fl} \leq \psi\left[x(t_f), u(t_f), p, t_f\right] \leq \psi_{fu} \tag{2.3}$$

where $\psi\left[x(t_f), u(t_f), p, t_f\right] \equiv \psi_f$. In addition the solution must satisfy *algebraic path constraints* of the form

$$g_{0l} \leq g\left[x(t_0), u(t_0), p, t_0\right] \leq g_{0u} \tag{2.4}$$

where g is a vector of size n_g, as well as simple bounds on the state variables

$$x_l \leq x(t) \leq x_u \tag{2.5}$$

and control variables

$$u_l \leq u(t) \leq u_u \tag{2.6}$$

Note that an equality contraint can be imposed if the upper and lower bounds are set equally a the same values. Finally it may be convenient to evaluate expressions of the form

$$\int_{t_0}^{t_f} q\left[x(t), u(t), p, t\right] dt \tag{2.7}$$

which involve the quadrature functions q. Collectively we refer to those functions evaluated during the phase, namely

$$F(t) = \begin{bmatrix} f\left[x(t), u(t), p, t\right] \\ g\left[x(t), u(t), p, t\right] \\ q\left[x(t), u(t), p, t\right] \end{bmatrix} \tag{2.8}$$

as the vector of continuous functions. Similarly functions evaluated at a specific point, such as boundary contidions, are referred to as *point functions*. The basic optimal control problem is to determine the n_u^k-dimensional control vectors $u^k(t)$ and parameter p^k to minimize the performance index

$$\mathcal{J} = \phi\left[x(t_0^1), t_0^1, x(t_f^1), p^1, t_f^1, \ldots, x(t_0^N), t_0^N, x(t_f^N), p^N, t_f^N\right] \tag{2.9}$$

Notice that the objective function may depend on quantities computed in each of the N phases.

This formulation raises a number of points which deserve further explanation. The concept of a phase, also referred to as an arc by some authors, partitions the time domain. In this formalism the differential equations cannot change within a phase, but may change from one phase to another. An obvious reason to introduce a phase is to accomodate changes in the dynamics, for example when simulating a multi-stage rocket. The boundary of a phase is often called an *event* or a *junction point*. A boundary condition which uniquely defines the end of a phase is sometimes called an *event criterion*. Normally the simulation of a complicated trajectory may link phases together by forcing the state to be continuous. If the objective function written in terms of quantities evaluated at the ends of the phases this is referred to as the *Mayer* form. If the objective function only involves an integral it is referred to as a problem of *Lagrange*, and when both terms are present it is called a problem of *Bolza*, as following:

$$\min_{x(t),u(t)} \mathcal{J}(t_0, t_F, x(t_0), x(t_F)) + \int_{t_0}^{t_F} w(\tau, x(\tau), u(\tau)) d\tau \tag{2.10}$$

2.1 Numerical Methods for Trajectory Optimization

Most methods for solving trajectory optimization problems can be classified as either *direct* or *indirect*. The key feature of a direct method is that it discretizes the trajectory optimization problem itself, typically converting the original trajectory optimization problem into a non-linear program. This conversion process is known as *transcription* and it is why some people refer to direct collocation metrhods as direct transcription methods. In general, direct transcription methods are able to discretize a continuous trajectory optimization problem by approximating all of the continuous functions in the problem statement as polynomial splines. Polynomials are used because they have two important properties: they can be represented by a small (finite) set of coefficients, and it is easy to compute integrals and derivatives of polynomials in terms of these coefficints. There are other direct collocation technqiques: direct single shooting, direct multiple shooting, orthogonal collocation, whose treatment

is out of the scope The *indirect methods* for solving trajectory optimization work by constructing analitically the necessary and sufficient conditions for optimality, and then solve them numerically.

2.1.1 Trapezoidal collocation

Let introduce a linear dynamical system that model a unit point mass that slides without friction in one dimension. The state of the block is its position x and velocity v, and the control is the force u applied to the block:

$$\dot{x} = v \tag{2.11}$$

$$\dot{v} = u \tag{2.12}$$

Next we need to write the boundary constraints which describe the initial and final state of the block. Here we constrain the block to move from $x = 0$ at time $t = 0$ to $x = 1$ at time $t = 1$. Both the initial and final velocity are constrained to be zero

$$x(0) = 0 \tag{2.13}$$

$$v(0) = 0 \tag{2.14}$$

$$x(1) = 1 \tag{2.15}$$

$$v(1) = 1 \tag{2.16}$$

A trajectory that satisfies the system dynamics and the boundary conditions is said to be feasible, and the corresponding controls are said to be *admissible*. We already said that a trajectory is optimal if it minizes an *objective function*. In genereal we are interested in finding solution trajectories that are both feasible and optimal. Here we will use a common objective funtion: the integral of control effort squared. This cost function is desidereable because it tends to produce smooth solution trajectories that are easily computed:

$$\min_{u(t),x(t),v(t)} \int_0^1 u^2(\tau) d\tau \tag{2.17}$$

We convert now the original continuous-time problem statement into a discrete form, obtaining a finite set of decision variables. This is done by representing the

continuous position $x(t)$, velocity $v(t)$ by their values at the *collocation points* (or phases)

$$t \rightarrow t_0 \dots t_N \tag{2.18}$$

$$x \rightarrow x_0 \dots x_N \tag{2.19}$$

$$v \rightarrow v_0 \dots v_N \tag{2.20}$$

$$\tag{2.21}$$

Next we need to convert the continuous system dynamics into a set of constraints that we can apply to the state and control at the collocation points. An usual discretization scheme is the *trapezoidal quadrature*. The key idea is thate the change in state between two collocation points is equal to the integral of the system dynamics. Thata integral is the approximated using trapezoidal quadrature, as shown below, where $h_l \equiv (t_{k+1} - t_k)$

$$\dot{x} = v \tag{2.22}$$

$$\int_{t_k}^{t_{k+1}} \dot{x} dt = \int_{t_k}^{t_{k+1}} v dt \tag{2.23}$$

$$x_{k+1} - x_k \approx \frac{1}{2}(h_k)(v_{k+1} + v_k) \tag{2.24}$$

Symplifing and then applying this to the velocity equation as well, we arrive at a set of equations that allow us to approximate the dynamics between each pair of collocation points. These constraints are known as *collocation constraints*. These equations are enforced on every segment: $k = 0 \dots (N-1)$ of the trajectory.

$$x_{k+1} - x_k \approx \frac{1}{2}(h_k)(v_{k+1} + v_k) \tag{2.25}$$

$$v_{k+1} - v_k \approx \frac{1}{2}(h_k)(u_{k+1} + u_k) \tag{2.26}$$

Finally, we approximate the objective function using trapezoid quadrature, converting it into a summation over the control effort at each collocation point

$$\min_{u(t)} \int_{t_0}^{t_N} u^2(\tau) d\tau \approx \min_{(u_0,\dots,u_N)} \sum_{k=0}^{N-1} \frac{1}{2}(h_k)(u_{k+1}^2 + u_k^2) \tag{2.27}$$

The direct collocation methods transcribe a continuous-time trajectory optimization problem into a *non-linear program* (NLP), that is a constrained parameter optimization problem that has non-linear terms in either its objective or constraint function. A typical formulation for a non linear program is give :

$$\min_{z} \mathcal{J}(z) \tag{2.28}$$

`subject to:`

$$f(z) = 0 \tag{2.29}$$

$$g(z) < 0 \tag{2.30}$$

$$z_{low} \leq z \leq z_{upp} \tag{2.31}$$

In some cases, a direct collocation method might produce either a linear or quadratic program instead of a non-linear program. This happens when the constraints (including system dynamics) are linear and the objective function is linear (linear program) or quadratic (quadratic program). Both linear and quadratic programs are much easier to solve than non-linear programs, making them desirable for real-time applications, especially in robotics.

2.2 Discrete time optimal control formulation

Given the system model and constraints, a generic discrete time optimal control problem can be formulated as the following constrained NLP

$$\min_{x_0,u_0,\ldots,x_N,u_N} \sum_{k=0}^{N-1} C(x_k,u_k) + C_N(x_N) \tag{2.32}$$

$$x_{k+1} - f(x_k,u_k) = 0 \tag{2.33}$$

$$h(x_k,u_k) \leq 0 \tag{2.34}$$

$$r(x_0,x_N) = 0 \tag{2.35}$$

where C is a function of the N-1 decision variables and C_N is a function of final decision variables.

We remark that other optimization variables could be present as well, such as a free parameter p that can be chosen bu is constant over time. Such parameters could ben added to the optimization formulation above by defining dummy states p_k that satisfy the dummy dynamic model equations

$$p_{k+1} = p_k \tag{2.36}$$

Note that the initial value of p_0 is not fixed by these constraints and thus we would have obtained our aim of having a time constant parameter vector that is free for optimisation.

This nonlinear program is large an structured and can thus be solved by any NLP solver. This is called the simultaneous approach to optimal control and requires the use of a structure exploiting NLP solver in order to be efficient. Note that in this approach, all original variables remain optimization variables of the NLP. Its name come from the fact that the NLP solver has to simultaneously solve both, the simulation and the optimization problem.

On the other hand, we know that we could eliminate nearly all states by a forward simulation, and in this way we could reduce the variable space of the NLP. The idea is to keep only x_0 and $U = [u_0^T, \ldots, u_{N-1}^T]$ as variables. The states $x_1, ..., x_N$ are

eliminated recursively by:

$$\bar{x}_0(x_0, U) = x_0 \tag{2.37}$$

$$\bar{x}_{k+1}(x_0, U) = f(\bar{x}_k(x_0, U), u_k) \tag{2.38}$$

then the optimal control problem is equivalent to a reduced problem with less variables, namely the following nonlinear program

$$\min_{x_0, U} \sum_{k=0}^{N-1} C(\bar{x}_k(x_0, U), u_k) + C_N(\bar{x}_k(x_0, U)) \tag{2.39}$$

$$s.t.: \tag{2.40}$$

$$h(\bar{x}_k(x_0, U), u_k) \leq 0 \tag{2.41}$$

$$r(x_0, \bar{x}_k(x_0, U)) = 0 \tag{2.42}$$

This reduced problem can now be addressed by a Newton-type methods, but the exploitation of sparsity in the problem is less important. This is called the *sequential* approach, because the simulation problem and optimization problem are solve sequentially, one after the other.

2.3 Karush-Kunh-Tucker condition of the discrete problem

First of all we review the Karush-Kuhn-Tucker Optimality conditions for the continuous problem. If x^* is a local minimizer of the NLP (2.28) and LICQ[1] holds, then exist so called multiplier vectors $\lambda \in \mathbb{R}^N$ with

$$\nabla f(x^*) + \nabla g(x^*)\lambda^* + \nabla h(x^*)\mu^* = 0 \tag{2.43}$$

$$g(x^*) = 0 \tag{2.44}$$

$$h(x^*) \leq 0 \tag{2.45}$$

$$\mu^* \geq 0 \tag{2.46}$$

$$\mu_i^* h_i(x^*) = 0 \tag{2.47}$$

[1] LICQ (Linear Independence Contrstrint Qualification) states that the gradients of the active inequality constraints and the gradients of the equality constraints are linearly independent at x^*

The KKT conditions are the First order necessary conditions for optimality (FONC) for constrained optimization. In the special case of convex problems, the KKT conditions are not only necessary for a local minimizer, but even sufficient for a global minimizer.

The Lagrangian function for the problem is defined as

$$L(x, \lambda, \mu) = f(x) + \lambda^T g(x) + \mu^T h(x) \tag{2.48}$$

Coming back to the discrete problem we have:

$$\min_w F(w) \tag{2.49}$$

$$s.t.$$

$$G(w) = 0 \tag{2.50}$$

where $G(w)$ is the vector of all constraints, and w is the vector of all decision variables.

$$G(w) = \begin{bmatrix} f(x_0, u_0) - x_1 \\ f(x_1, u_1) - x_2 \\ \dots \\ f(x_{N-1}, u_{N-1}) - x_N \\ r(x_0, x_N) \end{bmatrix}$$

The Lagrangian function for the discrete problem has the form

$$L(w, \lambda) = F(w) + \lambda^T\, G(w)$$

$$= \sum_{k=0}^{N-1} C(x_k, u_k) + C_N(x_N) + \sum_{k=0}^{N-1} \lambda_{k+1}^T (f(x_k, u_k) - x_{k+1}) + \lambda_r^T\, r(x_0, x_N) \tag{2.51}$$

and the summarized KKT-conditions of the problem are

$$\nabla_w\, L(w, \lambda) = 0 \tag{2.52}$$

$$G(w) = 0 \tag{2.53}$$

2.4 SQP methods for large nonlinear optimization

An efficient and reliable method that can solve large optimization problem with linear and non linear constraints is called SQP (Sequential Quadratic Programming). In general SQP is a class of methods that can divide and solve recursively a quadratic sub-problem (QP) at each iteration, classified in *major* and *minor* iteration. The major iteartion geneate a squence of iterates x_k that converge to the optimal x^* while the minor iteration is used to generate a search direction towards the next iterate x_{k+1}. In SQP the objective and constraint derivatives of (2.49) and (2.50) are required in order to define the objejctive and constraints of each QP sub-problem. In this thesis the SQP method is implemented by using the CasADi framework[2] that invole an Automatic Differentiation engine and a quadratic programming solver (IPOPT).

[2]http://www.casadi.org

Chapter 3

Optimal Trajectory Generation for quadrotors with DMOC

3.1 Introduction

Starting from the theory presented in the previous chapters we propose here to solve the trajectory optimization problem for quadrotor using DMOC (Discrete Mechanics and Optimal Control). The main advantage of this approach is to exploit the variational structure of the underlying mechanical system. The approach does not start from the derivation of the Euler-Lagrange equations for the system and uses a global discretization of the states and the controls borrowed by the discrete Lagrange-d'Alembert principle to obtain equality constraints for the discrete optimization problem. We will first present the Newton-Euler general model for a quadrotor, in order to introduce the phisical quantities involved, and then we will present the discrete optimization problem derived from the Hamilton-Pontryagin-d'Alembert principle.

3.2 Related Works

Discrete Mechanics and Optimal Control (DMOC) is an approach introduced in [36] by Junge et al., based on the direct discretizaion of the variational structure of a dynamic system. Discretizing the dynamic with a variational integrator [41], based on Lagrange-d'Alembert and Pontryagin principles, leads to structure preserving time-stepping equations which are used as equality constraints for Sequential Quadratic

Programming (SQP) methods. Theoretical bases of the approach was investigated by [51] [10] while application of the approach are traceable in [85] and [54].

3.3 The quadrotor Newton-Euler model

The following dynamic model of a quadrotor state for the case where the four rotors are *fixed pitch* and the thrust control is obtained through control of the torque of the motors. Let $I_f = (I_f^x, I_f^y, I_f^z)$ be a right-hand intertial frame, with E_z facing downwards. The vector x denote the position of the center of mass w.r.t. the frame I. Let $B = (B_f^x, B_f^y, B_f^z)$ denote a body fixed frame for the quadrotor. The orientations is given by a rotation $R : B \to I$ where $R \in SO(3)$ is an orthogonal rotation matrix. Let $v \in I$ denote the linear velocity expressed in the inertial frame and $\omega \in B$ denote the angular velocity of the airframe expressed in the body fixed frame. Let $\mathbb{J}$ denote the constant inertia matrix around the center of mass, and the mass matrix. Newton's equation of motion leads to the following model

$$\dot{x} = v \tag{3.1}$$

$$\boldsymbol{m}\dot{v} = \boldsymbol{m}\, g\, e_3 + R\, F \tag{3.2}$$

$$\dot{R} = R\, S(\omega) \tag{3.3}$$

$$\mathbb{J}\dot{\omega} = -\omega \times \mathbb{J}\omega + \Gamma \tag{3.4}$$

where the notation $S()$ denote the skew-symmetric matrix, the vector $F \in B$ combines the principal non-conservative forces applied to the quadrotors and $\Gamma \in B$ contains the differential thrust associated with pairs of rotors along with gyroscopic effect. The thrust applied to the airframe is

$$T = \sum_{i=1}^{4} f_i = k_t(\sum_{i=1}^{4} \bar{\omega}_i^2) \tag{3.5}$$

where $\bar{\omega}_i$ is the i-th motor angular speed is the k_t is a proportionality constant depending on the density of the air, the cube of the radius of the rotor blades, the

number of blades, the chord lenght of the blades, the lift constant (angle of attack). We can introduce a vector $\tau_b = (\tau_b^1, \tau_b^2, \tau_b^3)$ where:

$$\tau_b^1 = dk_t(\bar{\omega}_2^2 - \bar{\omega}_4^2) \tag{3.6}$$

$$\tau_b^2 = dk_t(\bar{\omega}_1^2 - \bar{\omega}_3^2) \tag{3.7}$$

$$\tau_b^1 = k_m(\bar{\omega}_2^2 + \bar{\omega}_4^2 - \bar{\omega}_1^2 - \bar{\omega}_3^2) \tag{3.8}$$

$$\tag{3.9}$$

where d is the distance of the i-th motor from the center of mass.

For our purpose the following model is used

$$\dot{x} = v \tag{3.10}$$

$$\dot{v} = g\, e_3 - \frac{1}{\boldsymbol{m}} T\, Re_3 \tag{3.11}$$

$$\dot{R} = R\, S(\omega) \tag{3.12}$$

$$\mathbb{J}\dot{\omega} = -\omega \times \mathbb{J}\omega + \tau_b \tag{3.13}$$

and we can collect the force and torque input using the mapping:

$$\begin{bmatrix} T \\ \tau_b^1 \\ \tau_b^2 \\ \tau_b^3 \end{bmatrix} = \begin{bmatrix} -k_t & -k_t & -k_t & -k_t \\ 0 & dk_m & 0 & dk_m \\ dk_m & 0 & dk_m & 0 \\ k_t & -k_t & k_t & -k_t \end{bmatrix} \begin{bmatrix} \bar{\omega}_1^2 \\ \bar{\omega}_2^2 \\ \bar{\omega}_3^2 \\ \bar{\omega}_4^2 \end{bmatrix} = \boldsymbol{A_L} \begin{bmatrix} \bar{\omega}_1^2 \\ \bar{\omega}_2^2 \\ \bar{\omega}_3^2 \\ \bar{\omega}_4^2 \end{bmatrix} \tag{3.14}$$

where $\boldsymbol{A_L}$ is called *control matrix*.

3.4 The optimization problem

In this section we introduce the discrete optimization problem for quadrotors in SE(3) following the work of Marsden and Kobilarov [41][54][42]. Let the configuration space be a Lie Group G with algebra $\mathfrak{g}$ and Lagrangian $\mathcal{L} : TG \to \mathbb{R}$ that is

left invariant under the action of G. Using the invariance we can left-trivialize such systems by introducing the body-fixed velocity $\xi \in \mathfrak{g}$ and the reduced Lagrangian $l : TG/G \to \mathbb{R}$ such that $l(\xi) = \mathcal{L}(g^{-1}g, g^{-1}\dot{g}) = \mathcal{L}(e, \xi)$. The system is required to move from a fixed initial state $(g(0), \xi(0)$ to a fixed final state $(g(T), \xi(T))$ during a time interval $[0, T]$ under the influence of a body-fixed control force $f(t) \in \mathfrak{g}^*$ (i.e. an internal force produced by actuators in the body reference frame) while minimizing:

$$\mathcal{J}(g, \xi, f) = \int_0^T C(g(t), \xi(t), f(t))\, dt \tag{3.15}$$

where C is a given cost function.

3.4.1 Lagrange d'Alembert-Pontryagin Principle

The Lagrange d'Alembert-Pontryagin (LDAP) principle introduced by Kobilarov is a generalization of the Lagrange-d'Alembert variational principle that provides additional freedom in the choice of variations which turns out to be crucial for obtaining symmetry and group structure preserving integrators and solving optimal control problems using this integrators. Let define the reduced path $(g, \xi, \mu) : [0, T] \to G \times \mathfrak{g} \times \mathfrak{g}^*$ and the control force $f : [0, T] \to \mathfrak{g}^*$. The principle requires that:

$$\delta \int_0^T [l(\xi) + \langle \mu, g^{-1}\dot{g} - \xi \rangle]\, dt + \int_0^T [T\, \mathcal{L}^*_{g^{-1}} f \cdot \delta g] dt = 0 \tag{3.16}$$

After taking variations the continuous equations of motion become:

$$\dot{\mu} - ad^*_{\xi}\mu = f \tag{3.17}$$

$$\mu = l'(\xi) \tag{3.18}$$

$$\dot{g} = g\, \xi \tag{3.19}$$

These equations are called the *Euler-Poincarè* equations and μ denotes the system momentum. The discrete version of the LDAP is:

$$\delta \sum_{k=0}^{N-1} \Delta t[l(\xi_k) + \langle \mu_k, \tau^{-1}(g_k^{-1}\, g_{k+1})/\Delta t - \xi_k \rangle] \tag{3.20}$$
$$+ \sum_{k=0}^{N-1} [T\mathcal{L}^*_{g_k^{-1}} f_k^- \cdot \delta g_k + T\mathcal{L}^*_{g_{k+1}^{-1}} f_k^+ \cdot \delta g_{k+1}] = 0$$

$$(d\tau^{-1}_{h\xi_k})^* \mu_k - (d\tau^{-1}_{-h\xi_{k-1}})^* \mu_{k-1} = \tilde{f}_k \tag{3.21}$$
$$\mu_k = l^{'}(\xi_k) \tag{3.22}$$
$$g_k^{-1}\, g_{k+1} = \tau(\Delta t \xi_k) \tag{3.23}$$

where $\tau : \mathfrak{g} \to G$ is the mapping from the algebra to the group.

The optimal control problem for a system with reduced Lagrangian $l : \mathfrak{g} \to R$ and fixed initial and final states can be directly formulated in discrete form as following:

$$\text{Compute:} \quad \xi_{0:N-1}, f_{0:N}, \Delta t \tag{3.24}$$
$$\text{Minimizing:} \quad J_d(g_{0:N-1}, \xi_{0:N-1}, f_{0:N}, \Delta t) = \textstyle\sum_{k=0}^{N-1} C_d(g_k, \xi_k, f_k, \Delta t) \tag{3.25}$$
$$\text{subject to:}$$
$$(d\tau^{-1}_{h\xi_k})^* \mu_k - (d\tau^{-1}_{-h\xi_{k-1}})^* \mu_{k-1} = \tilde{f}_k \tag{3.26}$$
$$\mu_k = l^{'}(\xi_k) \tag{3.27}$$
$$g_k^{-1}\, g_{k+1} = \tau(\Delta t \xi_k) \tag{3.28}$$

adding initial and final proper constraints for position, velocity and obstacles.

3.4.2 Derivation for quadrotors in SE(3)

In this section we present the discrete optimal trajectory problem for quadrotors modeled as a single underactuated body. The configuration space of the robot is $SE(3)$ and we use the control matrix $\mathbb{A}_{\mathbb{L}}$ in equation (3.14). The body fixed velocity ω and v corresponds to a Lie algebra element $\xi \in \mathfrak{se}(3)$ as in the following equation:

$$\xi = \begin{bmatrix} \omega_x & v \\ \mathbf{0}_{1\times 3} & 0 \end{bmatrix}$$

with ω_x the skew-symmetrix matrix

$$S(\omega) = \omega_x = \begin{bmatrix} 0 & -\omega^3 & \omega^2 \\ \omega^3 & 0 & -\omega^1 \\ -\omega^2 & \omega^1 & 0 \end{bmatrix}$$

The reduced lagrangian for the robot is:

$$l(\xi) = \frac{1}{2}(\omega^T \mathbb{J}\omega + v^T \mathbb{M} v) \tag{3.29}$$

A discrete version of the dynamics, e.g. a variational integrator in SE(3), is then employed to udate the position x_k, the orientation R_k and the velocities ω_k and v_k.

$$\begin{bmatrix} R_{k+1} & x_{k+1} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} = \begin{bmatrix} R_k & x_k \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \begin{bmatrix} \tau(\Delta t \omega_k) & \Delta t B_\tau(\Delta t \omega_k) v_k \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{3.30}$$

$$\begin{aligned} \Xi^T(\Delta t \xi_k) \begin{bmatrix} \mathbb{J}\omega_k \\ \mathbb{M} v_k \end{bmatrix} - \Xi^T(-\Delta t \xi_{k-1}) \begin{bmatrix} \mathbb{J}\omega_{k-1} \\ \mathbb{M} v_{k-1} \end{bmatrix} \\ = h \boldsymbol{A_L} u_k + \Delta t f_{ext}((R_k, x_k), (\omega_{k-1}, v_{k-1})) \end{aligned} \tag{3.31}$$

Where the map τ can be either defined as the classical exponential or the Cayley map (3.32), and consequently the Ξ matrix is (3.34).

$$cay(\omega) = \boldsymbol{I}_3 + \frac{4}{4 + \|\omega\|^2}(\omega_x + \frac{\omega_x^2}{2}) \tag{3.32}$$

$$B_{cay}(\omega) = \frac{2}{4 + \|\omega\|^2}(2\boldsymbol{I}_3 + \omega_x) \tag{3.33}$$

$$\Xi_{cay}(\xi) = \begin{bmatrix} \boldsymbol{I}_3 - \frac{1}{2}\omega_x + \frac{1}{4}\omega\omega^T & \mathbf{0}_3 \\ -\frac{1}{2}(\boldsymbol{I}_3 - \frac{1}{2}\omega_x)\hat{v} & \boldsymbol{I}_3 - \frac{1}{2}\hat{\omega} \end{bmatrix} \tag{3.34}$$

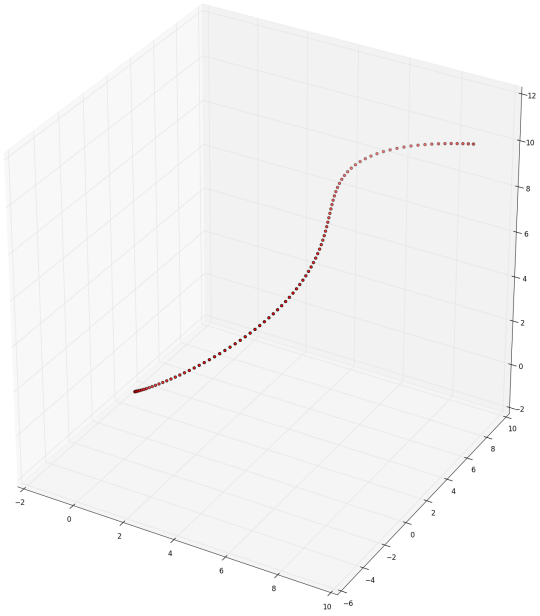
$$[ad_{(\omega,v)}] = \begin{bmatrix} \omega_x & \mathbf{0}_3 \\ v_x & \omega_x \end{bmatrix} \tag{3.35}$$

For better computational efficiency is possible to ignore the quadratic terms in the matrix (3.34), resulting in the map:

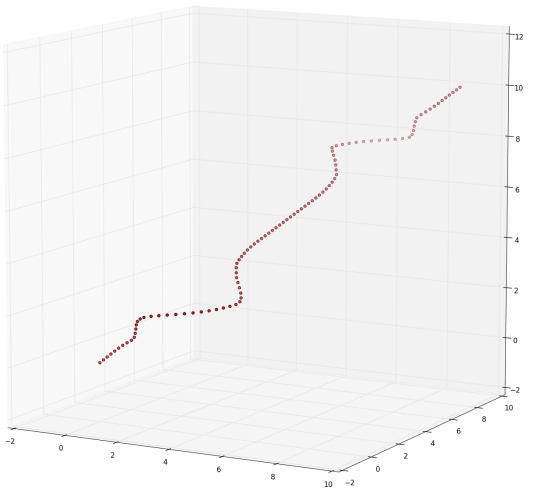
$$\Xi_{TLN}(\xi) = \boldsymbol{I}_6 - \frac{1}{2}[ad_{(\omega,v)}] \tag{3.36}$$

3.5 Implementation and Simulation Results

The optimal control problem (3.24-3.28) was transcripted using a *direct collocation* scheme in CasADi, that is an open-source tool for nonlinear optimization and algorithmic differentiation. We choose IPOPT (version 3.12.3) as nonlinear SQP solver. The equality constraints are written using (3.30) while the objective function is written using (3.31). The program, written in python, computes an optimal trajectory minimizing the control energy or the angular velocity, while avoiding two obstacles. The number of phases of the trajectory is $N = 100$ while the time horizon is $T = 5.8secs$. The quadrotor starts from $x_0 = (0,0,0)$ and stop in $x_f = (9,9,10)$. The problem consist of 2158 variables, 1470 equality constraints and 295 inequality constraints. The solver find out an admissible optimal solution in 99 iterations with an overal NLP error of $8.098e^{-9}$ in 3.264 secs on an Ubuntu Linux machine with Intel core-i7 and $8gb$ of ram.

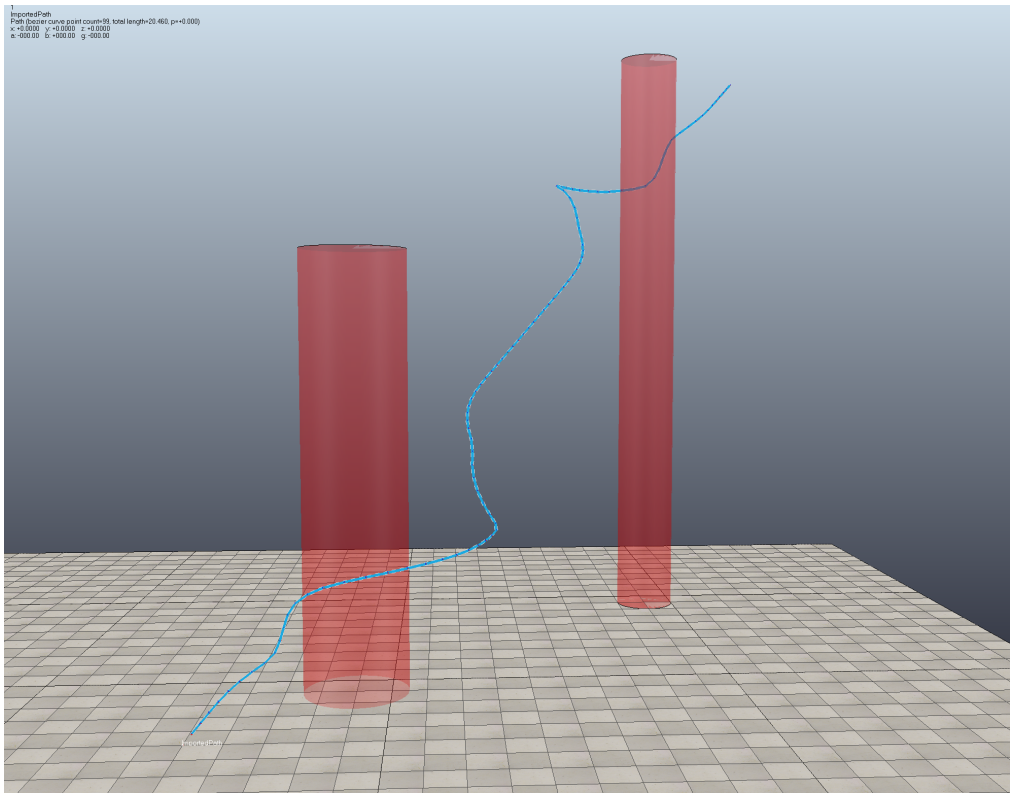


(a) Simulation results: optimal trajectory, minimization of the angular velocity

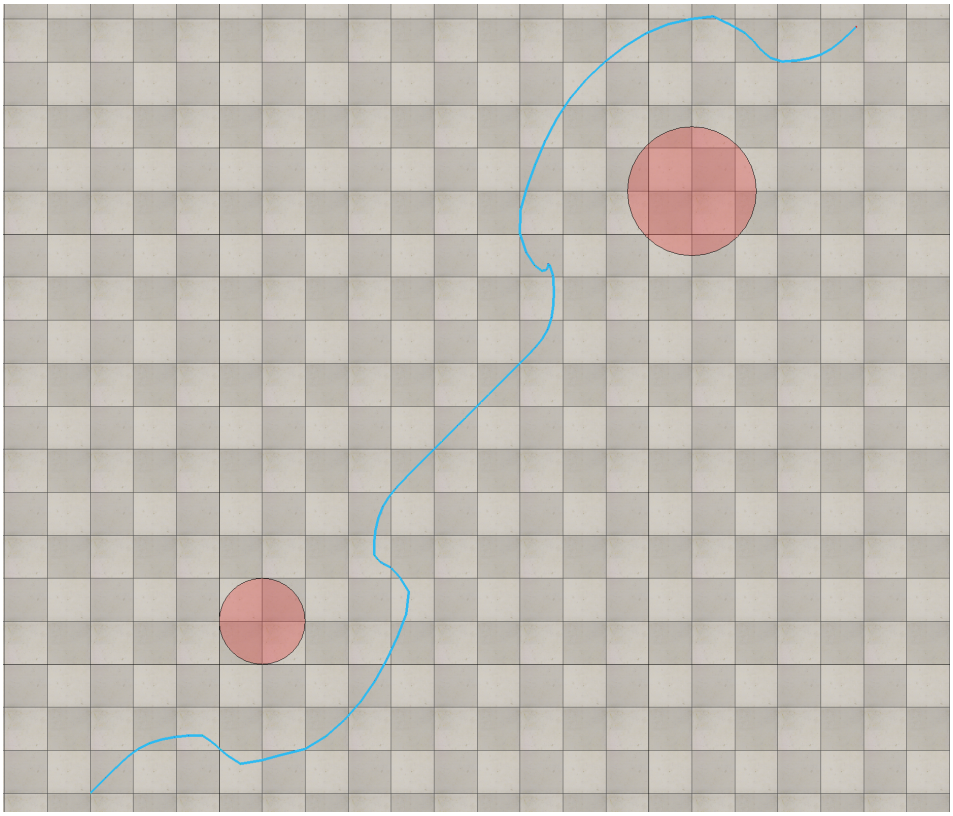


(b) Simulation results: optimal trajectory, minimization of the input energy

FIGURE 3.1: Trajectory simulation results

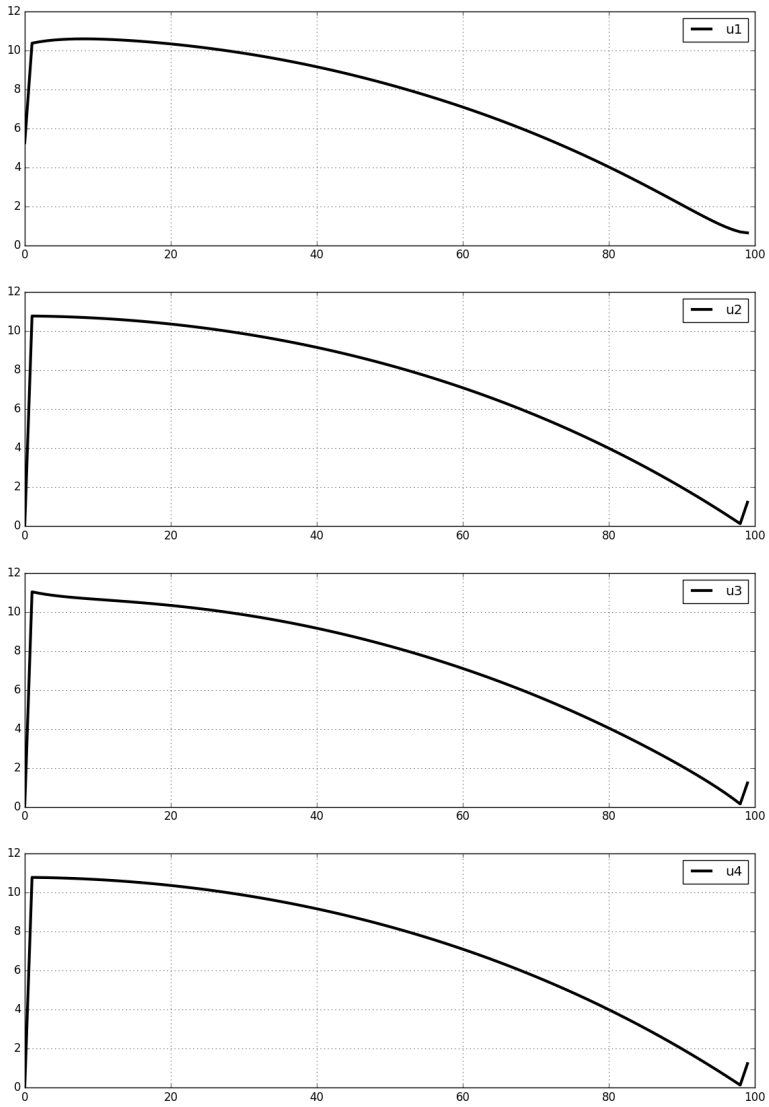


(a) Simulation results: trajectory exported in V-REP for realistic simulation purposes, 3d view

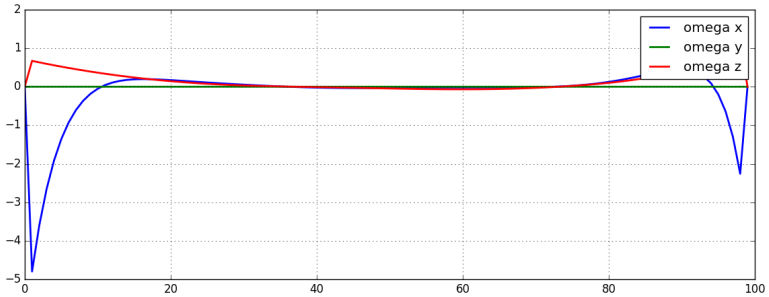


(b) Simulation results: trajectory exported in V-REP for realistic simulation purposes, from above

FIGURE 3.2: V-REP simulation environment



(a) Simulation results: computed control input for the case of angular velocity minimization



(b) Simulation results: angular velocity

FIGURE 3.3: Computed control signal and angular velocities

Chapter 4

Minimal information strategies for motion-planning in multi-floor navigation

This chapter presents the application of a minimalistic navigation strategy based on the well-known BUG2 algorithm, to solve the problem of reaching a goal position in an *unstructured* multi-floor indoor scenario using a quadrotor. Examples of this scenario include buildings and in general cluttered indoor areas. As far as energy backup is concerned the quadrotor shows stricts constraints: for this reason implementing a low-consumption navigation strategy is a major issue. We present a two-layer navigation strategy, called MF-BUG2, useful to navigate in multi-floor buildings starting from the ground floor toward the last or viceversa while searching for an interesting phisical quantity (i.e. gas leak, electromagnetic source). In the lower layer a BUG-like algorithm is able to drive the flying robot, equipped with a salient-cue sensor and a laser-range-finder, towards the estimated position of goal on the horizontal plane while avoiding obstacles and using minimal computational power and memory (the boundary-following behaviour uses an Artificial Potential Field to navigate around the obstacles). If the estimated goal position is reached but the salient-cue-sensor does not detect a salient quantity the higher level of the planner calls Dijkstra algorithm to computes the minimum-distance path to change the floor, assuming to know in advance the 2D position of the passages among different floors, and then moves vertically. The overall strategy is usefull for *indoor inspection* in hazardous scenarios. The algorithm is valited in simulation, investigating the

robustness with respect to the laser-range-finder noise.

4.1 Introduction

Research in the field of flying robots has shown a great potential, during the last decade, thanks to the extensive use of quadrotors [73]. The mechanical simplicity of this machine has encouraged a lot of researchers to develop autonomous or semi-autonomous flying robots able to navigate in outdoor and indoor scenario [60]. The applications of this technology ranges from agriculture to monitoring, from photogrammetry to Urban Search and Rescue (USAR). The main drawback lies in the small ability of carrying energy backup. This implies two main constraints:

1. on the sensor payload, due to the limited lifting thrust;
2. on the computational power, due to the limited current/hour capacity;

As a consequence, it is necessary to design navigation systems that uses minimal information and sensing in order to guarantee a low energy consumption. Surprisingly enough, the majority of the systems presented in the literature adopt an approach in designing quadrotor platforms that do not comply with constraints 1) and 2) above. Since the mainstream approach to navigation involves SLAM (Simultaneous Localization and Mapping) as a core part, heavy computational algorithms are needed to perform signal processing, features extraction, map building and path planning [77]. The nature of the *representation* of the world plays a fundamental role in this issue. A *global approach* provides the robot with the "most accurate" representation of the workspace considering the presence of incomplete data and sensor noise [58], while a *local approach* implements reactive, sensor-driven, navigation behaviours that takes inspiration from the biological beings [23] [87] [52]. It is worth noting that, referring to this framework, SLAM-based approaches can be considered *hybrid* because use a local information to build a global representation. A global approach is definitely not appropriate for our application, while it seems reasonable to adopt an hybrid approach built upon a local navigation strategy that satisfies technological constraints we are considering and allows to optimize the robot in terms of energy backup and miniaturization.

We presents the design of a two-layer hybrid navigation system for a quadrotor, called MF-BUG2, used to compute *on-line* and follow a path between a start and goal position in a multi-floor indoor scenario. The goal must be associated with some salient cue (visual, electromagnetical, olfactory [80]). The *lower layer* is based on the well know BUG2 algorithm that implements a local navigation approach, as discussed before, and is responsible of navigating in the horizontal plane. The *higher layer*, based on the Dijkstra algorithm, computes a sequence of subgoals (which are fed to the BUG2 algorithm in the lower layer) using the global information of a graph induced by the set W of 2D positions of the passages among the floors and is responsible of vertical navigation. Thereafter, assuming to navigate in a multi-floor building, the navigation system requires just the 2D position of the goal, the 2D position of the robot and the set W defined above. We assume to have a small quadrotor equipped with a 2D laser-range finder and a sensor able to sense the cue. The assumption about *indoor localization* is realistic thanks to UWB localization technology [24].

The work described is part of the research project PRISMA [1], aimed at intervention in post-disaster scenarios, and is usefull for *indoor inspection*, seeking for gas leaks on vertical pipes in multifloor buildings or other similar tasks.

The chapter is organized as follow. Section 4.2 presents the literature, focusing on the attempt to extend the minimal sensing problem to three-dimensional environments. Section 4.3 describes the navigation system, while Section 4.4 shows the results obtained in simulation. Conclusions follow.

4.2 Related Work

Navigation of robots with minimal information has been explored preeminently in 2D scenarios. The research community has been focused on extending BUG-like algorithms [52]. This class of navigation algorithms, thoughts for 2D point automatons and based on two possible behaviours (i.e. motion-to-goal, boundary-following) is huge and different classifications are present in the literature [58]. BUG2 needs to

[1] funded by the Italian Ministry of Research and Education

know the position of the automaton, the position of the goal (from which the distance is computed) and the information acquired from a contact sensor. An extension is TangentBug [38], which uses a range-sensor with a field-of-view of 360*deg* to acquire the distances from the obstacles around the robot and stores a set of *continuity intervals* in a data structure called Local Tangent Graph (LTG). Wedgebug algorithm [78] (and its extension RoverBug [78]) assumes to detect distances in a wedge. Ibug [81] uses an intensity sensor, which measures the signal strength emanating from the goal.

The first approach for solving the path planning problem for a mobile robot operating in an unknown three dimensional environment with minimal information was presented in [44] [45]. The authors discussed a crucial link between the path planning problem and the problem of visually exploring a three dimensional environment, concluding that visual exploration is necessary for succesfully planning the motion of a robot in unknown three-dimensional environments and introducing the notion of *exploratory algorithms*. Following this intuition Kamon and colleagues introduced 3DBug [37]. The robot is equipped with an ideal spherical range sensor with infinite detection range and the algorithm uses two modes of motion: motion-toward-the-target and obstacle-surface-trasversal (that is a generalization of boundary-following for 3D polytopial obstacles). The key point of the algorithm is the capability of expanding the knowledge of the obstacle, using range data, while attempting to reach an exit point along an obstacle surface. Due to the utilization of an ideal spherical sensor 3DBug is an interesting algorithm from a theoretical perspective. However, it is really hard to implement for its assumptions.

A review of the literature shows that the issue of navigating in a realistic 3D scenario with minimal sensing has not been successfully addressed without relaxing the energy constraint and assuming ideal sensors[77]. We start focusing our effort, following the purposes of project PRISMA, on the case of a multi-floor scenario.

4.3 Problem statement and system architecture

After an earthquake or other calamities there is the need to inspect indoor buildings, checking for the structural integrity, damages, unsafe situation or victims. The use

of robots allows to automate inspection tasks, helping rescuers in hazardous areas. Depending on the nature of the task the robot has to be equipped with a salient-cue-sensor to percieve an interesting phisical quantity (e.g. gas, light). In this cluttered scenarios it is unreasoneable to know in advance shape, dimensions and position of obstacles (e.g. walls, rubbles) while it is possible, looking at the planimetry, to estimate the position of vertical passages among different floors (e.g. stairwells) that could be eventually occluded. Thanks to research in the field of indoor localization it is today possible to localize a robot, also in cluttered areas, using different systems among which the most promising is based on Ultra Wide Band technologies [24]. Due to the three-dimensionality of the considered scenario a flying robot has a lot of advantages with respect to ground vehicles, and a quadrotor is the most versatile platform. A quadrotor is a rotorcraft that exploit the lift force provided by four rotors usually mounted in cross configuration. In order to define an orientation (or attitude) around its center of mass, three dynamic parameters, namely the angles of yaw, pitch and roll, are usually defined. The overall force F_B and torque M_B produced by the propellers, expressed in the body frame B, are used to control the quadrotor causing it to pitch, roll or yaw. By changing these three angles we are able to make the quadrotor maneuver in almost any direction. The quadrotor can move in the XY plane, it can change its altitude along the Z axis and can change its heading angle Φ (i.e. holonomic on the XY plane). Due to the fact that the quadrotor has no contact with the ground we do not use the Φ angle in our representation.

In order to formulate the algorithm let us introduce a number of definition. Let us define $S = (x_S, y_S) \in \mathbb{R}^2$ and $G = (x_G, y_G) \in \mathbb{R}^2$ as the Start and Goal locations of a mission, whereas $C = (x_C, y_C)$ is the current robot position. Let us define an *m-line* as the straight-line connecting S and G, and W as the vector of all the positions of the passages between floors, where each positions w_i, called waypoint, is expressed as $w_i = (x_i, y_i)$. Let us define a totally connected weighted graph $P = (N, A, M)$, with $N = W \cup G$ the initial set of nodes, A the set of bidirectional arcs, and a set of weights M containing the Eucledian distances between nodes. Finally, let us define $d\langle X, Y\rangle$ as a function which measures the Eucledian distance between any two points $X, Y \in \mathbb{R}^2$. We suppose to equip the quadrotor with two sensors: a laser-range-finder with a field of view of $360deg$ and a salient-cue sensor SC triggered by some

interesting quantity as specified above.

As mentioned above MF-BUG2 is a two-layer navigation system. The lower level exhibits three different behaviours: `motion-to-goal`,`boundary-following`, `moving-vertically`. The laser scanner is used to emulate a contact sensor, yet keeping the quadrotor at a safety distance from obstacles. During `motion-to-goal`, which is the starting beahavior, the robot moves along the m-line, until an obstacle. At this point the algorithm switches to `boundary following` behaviour and stores the Cartesian position of the *hit-point* H_j and the distance $d\langle H_j, G\rangle$. It keeps following the boundary of the obstacle until it cross again the m-line and the distance between the current position of the robot and the goal $d\langle C, G\rangle$ is smaller than $d\langle H_j, G\rangle$. At that point, a *leave-point* L_j is defined and the motion-to-goal behaviour is invoked again. The planner, as usual for the BUG2, tries to minimize the distance between the robot and the goal.

As soon as the estimated goal position is reached on the horizontal plane, if the salient-cue sensor is not triggered, the higher level of the navigation system invokes $Dijkstra(\langle C, W, G\rangle, A, M)$ to plan the shortest path on the induced weighted directed graph $P = (N, A, M)$ with an extended set of nodes $N = \langle C, W, G\rangle$ and edges A. The node C, representing the current position, is now connected to each node of W with simple directed arcs and each node of W is connected to G with a simple directed arc. Notes that there is no arcs between C and G. The Dijkstra algorithm returns the shortest path between C and G, composed of three nodes where the second node is the 2D position of *vertical passage* (waypoint), and then is set as a subgoal for BUG2, while the main goal position is stored in the variable *MainGoal*. As soon as this latter is reached the robot moves along Z of a constant altitude (`moving-vertically` behaviour) changing the floor while using an obstacle avoidance strategy and finally start again moving toward the main goal. The overall navigation behaviour makes the robot moving from a floor to the next one, reaching the estimated goal position at each floor, until it find the cue and then terminates. The proposed algorithm is intuitively $O(n)$ in space complexity where $n = N - 1$ and N is the number of floor of the building.

Algorithm 1 MF-BUG2

1: $W = ((x_1, y_1), \ldots, (x_n, y_n))$ ▷ Input: Waypoint
2: (S, G) ▷ Input: Start and Goal position
3: SC ▷ Input: Salient-cue sensor
4: $j \leftarrow 1$
5: $L_0 \leftarrow S$
6: `moving-vertically` $= 0$
7: **while** $S \neq C$ **do**
8: `motion-to-goal`
9: `Update C`
10:
11: **if** `Goal is reached` $\wedge\, SC = 1$ **then**
12: Exit
13: **end if**
14: **if** `Goal is reached` $\wedge\, SC = 0 \wedge$ `moving-vertically` $= 1$ **then**
15: Move along Z axis
16: $G \leftarrow MainGoal$
17: `moving-vertically` $\leftarrow 0$
18: **end if**
19: **if** `Goal is reached` $\wedge\, SC = 0$ **then**
20: `ShortesPathWaypOint` $\leftarrow$ `Dijkstra`$(\langle C, W, G\rangle, A, M)$
21: `MainGoal` $\leftarrow G$
22: $G \leftarrow$ `ShortestPathWaypoint`
23: `moving-vertically` $\leftarrow 1$
24: **end if**
25: **if** `Obstacle is encountered` **then**
26: $H_j = C$
27: `boundary-following`
28: **end if**
29: **while** $C \neq H_j$ **do**
30: **if** $C = G$ **then**
31: Exit
32: **end if**
33: **if** C is on the m-line **then**
34: **if** $d\langle C, T\rangle < d\langle H_j, T\rangle$ **then**
35: **if** the robot does not cros an obstacle at C **then**
36: $L_j = C$
37: $j \leftarrow j + 1$
38: Break
39: **end if**
40: **end if**
41: **end if**
42: **end while**
43: **end while**

The boundary following behaviour is implemented using an Artificial Potential Field, as used in *µnav* [58]. The robot motion law is given by:

$$v = [W_g(U)g + W_t(U)t + W_f(U)f]V_{ref} \tag{4.1}$$

where v is the velocity of the robot, V_{ref} is a reference value for the speed, g is a unit vector directed to the goal, t is a unit vector tangent to the APF equipotential line, f is a unit vector directed along the inverse gradient of the potential field, U is the current value of the potential field, and W_g, W_t, W_f are weights which depends on U.

4.4 Simulation Results

The simulations are performed in VREP[2] (Virtual Robot Experimentation Platform) that provides an integrate environment with a dynamic engine, 3D objects (robot, furnitures) and sensors. VREP allows to achieve a realistic physical simulation, including collisions with the environment and dynamic flight simulation of the rotorcraft. The navigation system is implemented from scratch in ROS[3], using the Point Cloud Library[4] (PCL) to acquire the laser scanner data.

Based on the model of a simple quadrotor, equipped with an Hokuyo URG-04LX laser-scanner and a radio-transciever (with 0.5 meter of range) as salient-cue sensor (fig. 4.1.a and 4.1.b), the validity of the proposed algorithm has been tested in a two-floors office-like cluttered virtual environment, showed in fig. 4.1.c. The walls have a tickness of 25 cm and some office fornitures (e.g. chairs, bookshelf) are present. A second radio-transciever is positioned at the goal, to trigger the salient-cue sensor. The position of vertical waypoint passages are manually inserted in the scenario (visualized as small red spheres). The start and goal positions are randomly selected. The `moving-vertically` behaviour is set to generate a vertical path segment of 2 meters avoiding vertical obstacles with a simple obstacle avoiding strategy. In all simulations the robot has reached the goal, changing the floor correctly. In Table 4.1

[2] http://www.coppeliarobotics.com
[3] http://www.ros.org
[4] http://www.pointclouds.org

we show a set of significant simulations that presents a different number of *Hit* and *Leave* point.

Moreover, we focused on testing robustness to laser-range-finder noise, simulating an AWGN noise (gaussian with zero mean) affecting distance measurements retrieved from VREP.

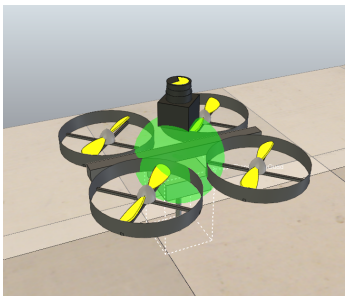
Table 4.2 and fig. 4.2.b show results, using as a reference the start and goal position of simulation number 9 of Table 4.1 (that has zero noise), and by iteratively adding noise with an increasing Standard Deviation of $\sigma = 0.025, 0.05, 0.1$. With $\sigma = 0.025$ and $\sigma = 0.05$ we do not have a significant change in travel-time and lenght, while with $\sigma = 0.1$ is observable a shortening of the path lenght due to the fact that the robot navigates closer to the obstacle.

Simulation	H	L	U	Lenght	RD	Time
1	0	0	1	14.74	0.29	1:29
2	2	2	1	22.50	0.26	2:14
3	2	1	1	17.39	0.29	1:29
4	4	4	1	16.62	0.28	2:17
5	2	1	1	22.08	0.29	2:11
6	2	2	1	24.11	0.24	3:44
7	3	3	1	24.75	0.27	4:32
8	3	3	1	40.21	0.29	4:46
9	2	2	1	19.00	0.27	2:35
10	5	5	1	23.06	0.28	1:56

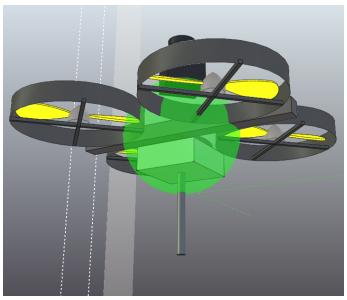
TABLE 4.1: H = num of hitpoints, L = num of leave points, U = num of floors passages, Lenght = path's lenght (meters), RD = Residual Distance to the goal (meters), Time = travel time. These measurements are taken without adding noise ($\sigma = 0$).

Simulation	σ	Lenght	Time
1	0.025	18.79	2:37
2	0.05	18:72	2:24
3	0.1	13:61	2:39

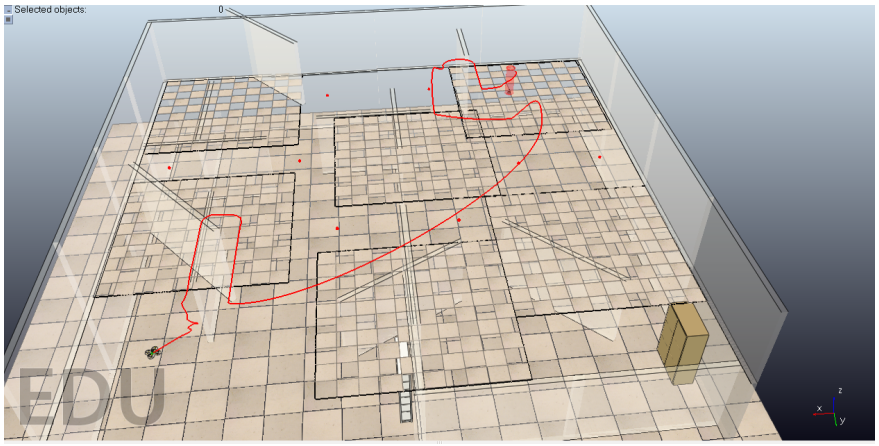
TABLE 4.2: Measurements relative to simulation 9 of Table 1 adding Gaussian Noise. σ = Standard Deviation of AWGN noise with $\mu = 0$, Lenght = path's lenght (meters), Time = travel time.



(a) Simulated quadrotor equipped with an Hokuyo Laser Scanner

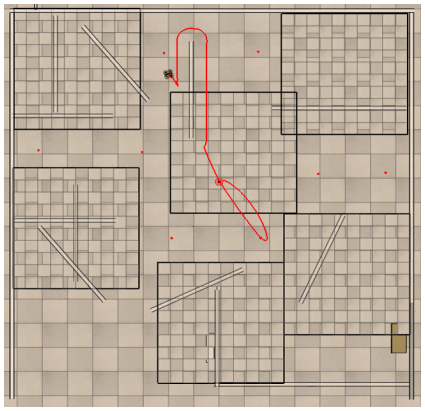


(b) Radio Salient-cue Sensor



(c) The simulated scenario represents an office-like building with two floors. The red dots are the position of the waypoint passages w_i. The planner activate the `moving-vertically` behaviour near the goal, showed as a red cilinder

FIGURE 4.1: V-REP simulation environment



(a) The red line is the path from Start to Goal. The yellow circle is the sensing radius of the salient-cue sensor

(b) Noisy trajectory with $\sigma = 0.1$ and $\mu = 0$

FIGURE 4.2: V-REP trajectories

4.5 Conclusions

The chapter presented a simple approach to plan the motion of a quadrotor in a indoor multi-floor scenario with minimal information. The navigation system is explicitly designed for autonomous inspection tasks in hazardous multi-floor buildings, assuming the availability of a wireless localization systems based on Ultra Wide Band[5] technology. Our design objective is to satisfy the energy constraint typical of quadrotors. The navigation system is based on the well known minimalistic BUG2 and Dijkstra algorithms. The robot requires only a laser-range-finder and a salient-cue sensor to capture interesting quantities for post-disaster indoor inspection. Experiments in simulation have shown that the proposed planner performs well, also in presence of noisy measurements, and can be used in real scenarios. Our future effort will be focused on implementing and testing the strategy using a real quadrotor.

[5] http://disal.epfl.ch/page-47308-en.html

Chapter 5

Quadrotor navigation with visual features using Pose-Based Visual Servoing

In this chapter an image-based control strategy is presented for the navigation of a quadrotor along a corridor-like indoor unstructured environment. The robot is supposed to be equipped with a frontal pinhole camera and a downward optical-flow sensor. We focused on a minimum set of image-plane features naturally extractable from the video sequence. This allow the robot to navigate in an unstructured, markerless, environment. The quadrotor fast dynamics are stabilized by model linearization and classical proportional/derivative controllers. The navigation task is accomplished using a visual-servoing control law using a partioned approach. The control strategy is validated by simulation.

5.1 Introduction

Flying robot navigation in indoor environments remains a challenging task. Many approaches was proposed based on different fundamental assumptions: presence or absence of artificial markers in the environment (*environment augmentation*) [90], use of *global* or *partial* model of the world [76]. Making this assumptions is a fundamental step that is strongly constrained by the computational and energitical capabilities of the robot. A quadrotor observes very poor capabilities from this point of view. Following this consideration we focused our attention on minimalistic navigation

strategies, implementable by light-weight sensors, like monocular cameras, and low-power onboard computational units.

In this preliminary study, we aim at navigate a straight corridor using a monocular pinhole camera fixed onto a quadrotor in frontward position and an optical-flow sensor pointing downward. Instead of use the available visual information in a "look-and-move" fashion we exploit techniques developed in the field of *visual servo control* to establish a visual-feedback control loop that will increase the overall accuracy of the system [11][12][33].

In literature, visual servoing approaches are historically classified into Pose-Based Visual Servoing (PBVS) and Image-Based Visual Servoing (IBVS) both aimed to regulate to zero a visual error e built upon a set of visual features, namely s. In the former approach the error signal is defined in the *image-space* and the visual features s are directly used, conversely in the latter approach the error signal is defined in the *task-space*, so that the visual features are used to estimate task-space quantities related to the pose of the end-effector. There are very few visual-servoing control design for Quadrotors and Aerial Robots in general, most of them belonging to the PBVS class [2], therefore avoiding the image jacobian formulation. The key challenge here is managing the highly coupled dynamics of the robot model, due to the underactuation. The IBVS experiments, to our knowledge, are mostly restricted to the case of a downward camera, using a pinhole or spherical camera model [91][25][56][62][83].

The visual servoing method proposed use as visual features both the *vanishing point* and the corridor *mid-point*, extracted from the video stream of a frontward monocular camera. This approach was already proposed for ground robot [86][21][65][15], so the main contribution of this chapter is the use for a flying robot.

The chapter is organized as follows. Section 5.2 concern the modelling of the quadrotors. Section 5.3 defines the visual features. Section 5.4 concerns the control strategy. Simulation results are presented and discussed in Section 5.5. The last section points out conclusions and future work.

5.2 Quadrotor Dynamic Model

A quadrotor is a VTOL (Vertical Take-Off and Landing) aerial vehicles, with four identical rotors and propellers located at the vertices of a square, that can be modeled as a rigid body immersed in a fluid and moving in a 3D space. Different models are considered depending on the flight condition under consideration. A very complete and general model, taking into account all the external torques and disturbances is presented in [32] and [31]. While these latter are based on a Newton-Euler formulation in [62] is presented a modelization based on Hamilton-Lagrange theory. We will start from the model presented in [47].

Let be $I = \{e_1, e_2, e_3\}$ an inertial reference frame and $B = \{b_1, b_2, b_3\}$ a body fixed frame. The origin of the model is located at the center of mass of the quadrotor.

Let define $m \in \mathbb{R}$ the total mass, $J \in \mathbb{R}^{3\times3}$ the inertia matrix with respect to the body-fixed frame, $R \in SO(3)$ the rotation matrix from the body frame to the inertial one parametrized by Euler angles as $\eta = (\phi, \theta, \psi)$ and referred as Roll, Pitch and Yaw respectively, $\Omega \in \mathbb{R}^3$ the angular velocity in the body frame, $\xi = (x, y, z) \in \mathbb{R}^3$ the position of the center of mass in the inertial frame, $v \in \mathbb{R}^3$ the velocity vector of the center of mass in the inertial frame, $f_i \in \mathbb{R}$ the thrust generated by the i-th propeller along the $-b_3$ axis, $f \in \mathbb{R}$ the total thrust defined as $\sum f_i$, $M \in \mathbb{R}^3$ the total moment vector in the body fixed frame.

The configuration of the quadrotor is defined by the position of its center of mass and the attitude with respect to I. So the configuration manifold is $SE(3) = \mathbb{R}^3 \times SO(3)$.

The equations of motion of the quadrotor can be written as

$$\dot{\xi} = v \tag{5.1a}$$

$$m\dot{v} = mge_3 - fRe_3 \tag{5.1b}$$

$$\dot{R} = R\hat{\Omega} \tag{5.1c}$$

$$J\dot{\Omega} + \Omega \times J\Omega = M \tag{5.1d}$$

where $\hat{\Omega}$ is the Skew-Simmetric matrix of Ω.

The control inputs are $u = (f, M) = (f, \tau_1, \tau_2, \tau_3)$ and the control over the motion

of the frame B is obtained by using the torque $M = (\tau_1, \tau_2, \tau_3)$ to reorient fRe_3 in the desired direction. For the purpose of this research we can avoid the Coriolis effect on the dynamic model simplifing the equations as following [70]:

$$m\ddot{x} = -f\ (cos(\psi)sin(\theta)cos(\phi) + sin(\psi)sin(\phi)) \tag{5.2a}$$

$$m\ddot{y} = -f\ (sin(\psi)sin(\theta)cos(\phi) - cos(\psi)sin(\phi)) \tag{5.2b}$$

$$m\ddot{z} = -f\ (cos(\theta)cos(\phi)) + mg \tag{5.2c}$$

$$\ddot{\phi} = \tau_1 \tag{5.2d}$$

$$\ddot{\theta} = \tau_2 \tag{5.2e}$$

$$\ddot{\psi} = \tau_3 \tag{5.2f}$$

The simplified model shows that from the underactuated overall systems we can isolate a nonlinear $x - \theta$ subsystem for the forward translational dynamics, a nonlinear $y - \phi$ subsystem for the lateral translational dynamics and ψ subsystem for the yaw dynamic.

5.3 Visual Features Definition

The control objective is to navigate at the center of a straight corridor using information from the camera, while mantaining a costant forward velocity $v = v_d$ in the body x direction. The strategy is to compute as visual features the *vanishing point* and the corridor *middle point*, and search for a relationship of both with the ψ and y dynamics. The first feature is defined as the intersection of the environment perspective straight lines, extracted from the video sequence using Hough Transform in each frame and a Gaussian sphere projection [61] of the extracted line segements. The second is the midpoint of the segment between the intersection points of the floor-wall boundaries edges with the horizontal axis of the image plane. The low-level image processing phase is out of the scope of this work.

Let define a frame C fixed onto the camera with the origin at the focal-points and $T_c = T_c(x, y, z, \phi, \theta, \psi)$ the homogeneous transformation from the inertial frame I and the camera-fixed frame C. The geometric relationship between any 3D points $X_0 = (X_0, Y_0, Z_0)^T$ and their corresponding image coordinates $\bar{x}_{im} = (x_{im}, y_{im})$ (in

pixels) depends on the pose captured by the 4×4 matrix T_c, on the perspective 3×4 projection matrix $P = (Id_3, 0^{3\times1})$ where Id_3 is a 3×3 identity matrix, and on the 3×3 *calibration matrix* $A = A(\lambda, \alpha_x, \alpha_y, o_p)$ where λ is the focal lenght, (α_x, α_y) are the horizontal and vertical scale factor, o_p is the principal point in the image plane.

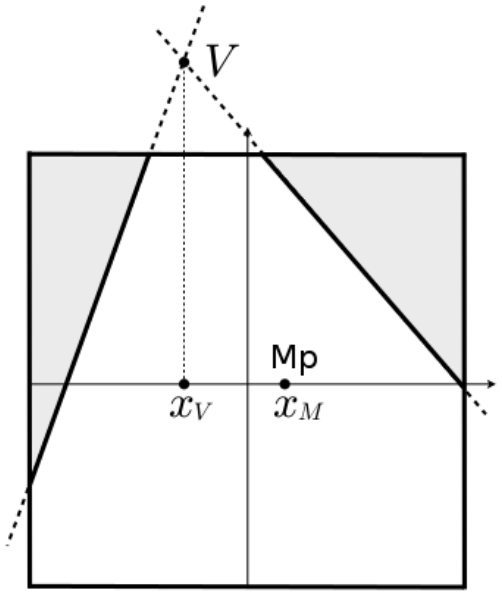


FIGURE 5.1: Typical image-plane view of a quadrotor in motion along a straight corridor [86]. V is the vanishing point, M_p is the corridor middle point. We are interested in regulate to zero the abscissas.

The *perspective projection model* [55] is expressed by

$$\lambda \bar{x}_{im} = APT_c X_0 \tag{5.3}$$

As showed in [5] the vanishing point belongs to the plane at infinity and its observed motion in the image plane depends only on the rotation part of the camera displacement. Following the IBVS approach [11][91] we need to establish a differentiable relationship between the image features and some degrees of freedom. Accordingly, the abscissas of the vanishing point and the middle point, using the projection model, can be computed as [21]:

$$x_V = k_1 \, tan(\psi) \tag{5.4a}$$

$$x_M = k_2 \, \frac{\tilde{y}}{cos(\psi)} + k_3 \, tan(\psi) \tag{5.4b}$$

where $\tilde{y} = y - d/2$ and d is the horizontal width of the corridor. Regulating to zero the x_V feature keep aligned the quadrotor to the corridor direction, while regulating to zero the x_M feature translate the quadrotor at the center of the corridor. Both the image feature are supposed to be always in the field of view of the camera. Assuming a constant tilt-angle β and $\beta \neq 0$ for the camera, the k_i parameters can be defined as $k_1 = \alpha_x \beta / cos\beta$, $k_2 = -\alpha_x \lambda sin\beta / z$ and $k_3 = \alpha_x \lambda cos\beta$.

Differentiating equations (4) leads to:

$$\dot{x_V} = \frac{x_V^2 + k_1^2}{k_1}\,\omega \tag{5.5a}$$

$$\dot{x_M} = \frac{k_2 x_V}{k_1}\,v + (k_3 + \frac{x_M x_V}{k_1})\,\omega \tag{5.5b}$$

Following this results is possible to derive a *partioned* visual-servo control law for the servoing of the Yaw dynamics during a constant velocity forward flight, as shown in the next section.

5.4 Control Strategy

Besides the frontal monocular camera the quadrotor is supposed to be equipped with an Inertial Measurement Unit (IMU) and a downward optical-flow sensor joined with an ultrasonic range sensor [30]. This sensor arrangement allow to estimate the vehicle's state: $z, \dot{x}, \dot{y}, \dot{z}$ are measured with the optical-flow and ultrasonic sensor, while y, ψ are estimated using visual information by means of equations (5.4). The IMU measures the angular rate $\dot{\phi}, \dot{\theta}, \dot{\psi}$ and estimate the Euler angles ϕ, θ by sensor fusion.

The z position and the Yaw velocity $\dot{\psi}$ can be stabilized using a proportional/derivative controllers

$$f = m(-k_{dz}\dot{z} - k_{pz}e_z + g)\,\frac{1}{cos(\theta)cos(\phi)} \tag{5.6a}$$

$$\tau_3 = -k_{d\psi}\dot{\psi} + k_{p\psi}\dot{\psi}_d \tag{5.6b}$$

where $e_z = z_d - z$ and z_d is the desired altitude. Substituing $(5.6a), (5.6b)$ in equations (5.2) with $cos(\theta)cos(\phi) \neq 0$, and considering that for a large enough time the

errors e_z, e_ψ become arbitrary small, the translational dynamics model reduces to:

$$\ddot{x} = -g\ tan(\theta) \tag{5.7a}$$

$$\ddot{y} = g\ \frac{tan(\phi)}{cos(\theta)} \tag{5.7b}$$

Let consider now the $x - \theta$ subsystem given by equation $(5.7a)$ and $(5.2e)$. For small angle we obtain $\theta \approx tan(\theta)$ and substituting we obtain a linearization of the system.

$$\ddot{x} = -g\ \theta \tag{5.8a}$$

$$\ddot{\theta} = \tau_2 \tag{5.8b}$$

In order to stabilize the pitch angular position and to obtain a forward flight at constant velocity $\dot{x_d} = v_d$ the authors in [86] introduces the following PD control law:

$$\tau_2 = -k_{p\theta}\theta - k_{d\theta}\dot{\theta} + k_{p\theta}k_{dx}\dot{x}_d - k_{p\theta}k_{dx}\dot{x} \tag{5.9}$$

Following the same procedure we can simplify the $y - \phi$ subsystem

$$\ddot{y} = -g\ \phi \tag{5.10a}$$

$$\ddot{\phi} = \tau_1 \tag{5.10b}$$

and stabilize the system to obtain a zero lateral position with the following control law:

$$\tau_1 = -k_{p\phi}\phi - k_{d\phi}\dot{\phi} - k_{p\phi}k_{py}y - k_{p\phi}k_{dy}\dot{y} \tag{5.11}$$

Equation $(5.6b)$ allows to impose a desired angular velocity $\dot{\psi}_d = \omega$ that can be computed using the IBVS framework upon equations $(5.5a, 5.5b)$, while mantaining a forward small constant velocity along the body x axis, finally regulating to zero the visual features and navigating along the corridor. In [86] is showed the complete derivation of the partioned IBVS control law:

$$\omega = \frac{k_1}{k_1k_3 + x_mx_v}(-\frac{k_2x_v}{k_1}v_d - k_px_m) \tag{5.12}$$

that is of the form [11]:

$$v_\omega = -L_\omega^+(k_p e(t) + L_v \nu) \tag{5.13}$$

where is introduced k_p as a positive design constant. A Lyapunov-based stability proof for the control law is carried out in [86].

5.5 Simulation

Matlab/Simulink was used to perform a simulation of the closed-loop system using the linearized quadrotor model and the IBVS control law in (12). The translational dynamics are stabilized and regulated using the PD controller in (5.9) and (5.11) and is supposed flying at a constant altitude $z = 1\ m$ with a forward constant velocity $v_d = 0.5 m/s$, starting from position $(x, y) = (0, 0.5)\ m$ and with a initial yaw angle $\psi = 1\ rad$.

The PD parameters was set, according with [70], as following: $k_{p\theta} = 301.5$, $k_{d\theta} = 23.97$, $k_{p\phi} = 471.1$, $k_{d\phi} = 29.96$, $k_{p\psi} = 10$, $k_{d\psi} = 1$.

The visual parameters k_i was set assuming a frontal pinhole camera with $\lambda = 8\ mm$, $\alpha_x = 79.2\ pixel/mm$ and $\beta = 45\ deg$, resulting in $k_1 = 87.94$, $k_2 = -447$ and $k_3 = 447$.

The simulation shows the convergence of the approach.

Fig. 5.2 is the evolution over time of the control inputs, fig. 5.3 and 5.4 show an asimptotic behaviour for the Yaw ψ angle and y position over time, while fig. 5.5 show the $x - y$ trajectory of the robot composed of an initial smooth curve and a final straight path.

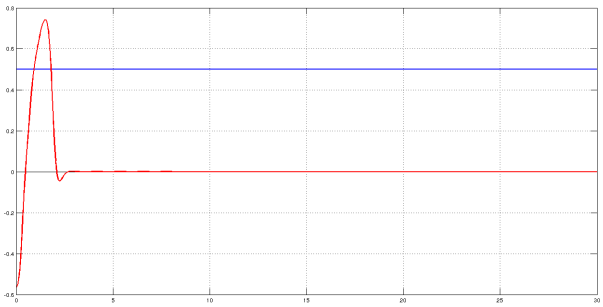


FIGURE 5.2: Simulation results (control input): ω in red, v_d in blue

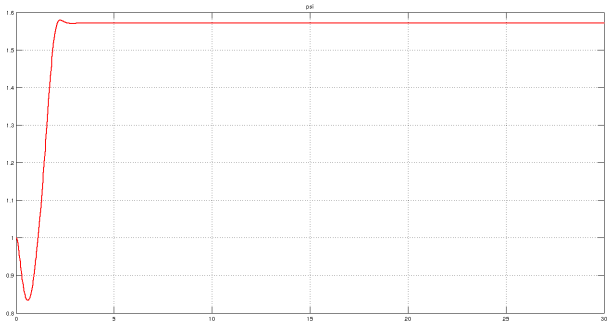


FIGURE 5.3: Simulation results: Yaw angle evolution (ψ)

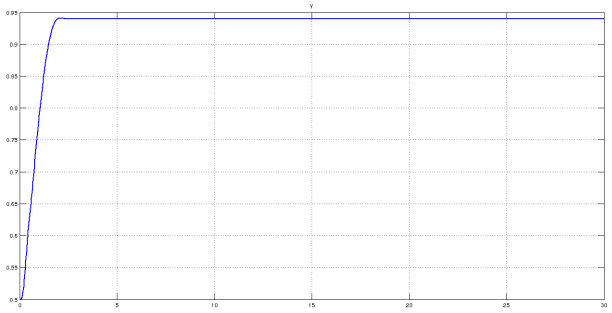


FIGURE 5.4: Simulation results: y position

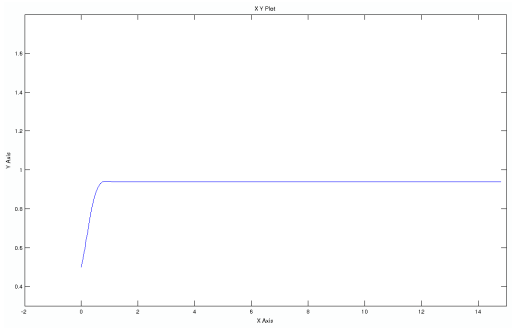


FIGURE 5.5: Simulation results: $x - y$ trajectory. The quadrotor turns with a smooth curve towards the center of the corridor and pursue along a straight trajectory

5.6 Conclusion

In this chapter it has been presented an image-based controller for the visual servoing of a quadrotor navigating in an unstructured corridor-like scenario with a frontal camera and a downward optical-flow sensor. Vanishing point and corridor mid-point extracted from the video stream was selected as visual features. A complete quadrotor model derivation and linearization was presented. A fast dynamics

stabilization based on proportional/derivative controllers was used to assign desired yaw behaviour by a partitioned visual-servoing control law. The strategy was proved in simulation and is susceptible of implementation on a real robot.

Chapter 6

Spatial reasoning and motion planning with Temporal Logic for micro-quadrotor

In this chapter we propose a multi-layer framework for distributed micro-quadrotor multi-agent planning from LTL specifications. We define for each robot a *local* LTL specification built upon a common set of Atomic Proposition. The High-Level planner compute a plan as a sequence of labeled waypoints and accept knowledge update from on-board sensors or neighbors agents. The Low-Level planner get the high level plan and compute feasible trajectories in SE(3) using a sampling-based algorithm. This planner checks the validity of randomly sampled states against an Octree-based Space Partition (OSP) that stores information about the workspace and allows collision-detection between agents. We validated the framework in simulated and real environment, using two 10cm micro quadrotors and a motion capture system.

6.1 Introduction

Recently, multi-agent systems have been gaining increasing attention especially to deal with tasks that can be done effectively by a group of robots such as patrolling, inspecting and search and rescue. Since we work with small and low-power robots, decentralization is advisable for scalability. In this chapter we address the problem

of collision-free navigation of multiple robots flying in a three-dimensional environment. In literature the problem is tackled, almost always in two dimensions, from different sides. The control community has produced a big literature on Consensus [71] and Reciprocal Obstacle Avoidance [6] while the planning community has developed a computational approach, based on multi-layer frameworks [7][9]. This work develops an approach based on automata and sampling-based planning and challenge the problem of *integrating task and motion planning* (ITMP)[29]. The main goal of ITMP is to integrate an high-level task specification with the continous low-level trajectory generation. The high-level task can be specified in different ways and could take or not into account the *temporal constraints* specified in a Temporal Logic language. The lower level is usually realized by Optimal Trajectory Generation [79]. Popular ITMP frameworks are based on two or three-layer with the high level planner based on linear temporal logic (LTL) language, an intermediate interface layer and a low-level planner. To the best of our knowledge we are presenting the first implementation of a three-layer ITMP architecture with sampling based low-level planner in SE(3) for a group of mobile robots.

6.2 Problem Formulation

Let W be a three-dimensional workspace that contains a finite set of obstacles and a finite set of *locations of interest (LOI)* V. The free workspace is called W_{free}. To each LOI is attached a standard label from the set $L = (l_1, l_2, \ldots, l_n)$ or an additional label from the set O, with $AP = L \cup O$. Let be $R_i = (R_1, R_2 \ldots R_N)$ a group of robot moving in the workspace W with unique identifiers $ID \in (1, 2 \ldots, N)$. Each robot has a *prior map* of the environment, coded in Finite Transition System (FTS). Each robot is also able to update the prior map with knowledge gained from sensors or by communicating with other agents.

6.2.1 Finite Transition Systems and Linear Temporal Logic

A Finite Transition Systems [84] [43] T is a mathematical object that model a discrete systems. It consists of states and transitions between states. It can be represented by a tuple $T = (S, I, Act, G)$ where S is a finite set of states; I is a set of initial states; Act

is a set of binary relations between states; G is a set of Goal states. A transition system is *deterministic* if there is only one initial state and all actions are deterministic, and in this case all future states are predictable. A Nondeterministic Buchi Automaton (NBA) is a non-deterministic FTS that accepts as input infinite sequences.

Linear Temporal Logic[66] is a formal logic that extends classical propositional logic to include operators referring to time. In LTL it is possibile to encode formulae about future or past states, e.g. conditions that will be *eventually true* or *always true*. Let P be a set of *Atomic Proposition*. Let also introduce the operators: $\Box$ Always, $\Diamond$ Eventually and U Until. A formula $\phi_1 U \phi_2$ means that ϕ_1 has to be true until ϕ_2 becomes true, a formula $\Box\phi_1$ means that ϕ_1 has to be always true in future states, while $\Diamond\phi_1$ means that a formula has to be true at some future state. Every formula is evaluated over a sequence of states $\sigma = \sigma_0\sigma_1 \ldots \sigma_n$ using an interpretation function $\sigma \colon \mathbb{N} \rightarrow 2^P$. For every LTL formula ϕ there exists a Nondeterministic Buchi Automaton (NBA) A^ϕ, defined as $A^\phi = (Q, 2^P, \delta, Q_0, F)$ where Q is a set of states; Q_0 is a set of initial states; δ is a set of transition and F is a set of accepting states. A *run* is a sequence of states produced, on an in input sequence, by the automata. The Buchi Automata is checking automata used to recognize paths that satisfy the LTL specification. For an insight about LTL checking refer to [84].

6.2.2 Octree Space Partitioning

One of the novelty of this work is given by the use of an Octree Space Partition (OSP) to store obstacles and trajectories computed by neighbors agents. Each agent has a local Octree and a set of primitives to work with. An *Octree* is a recursive, hierarchical partition scheme that uses a tree structure to represent a volume in the space and is formed by recursive subdivision. Such a representation [75] is commonly used in computer graphics to optimize collision detection or nearest neighbor search. The space is recusively subdivided in eight disjoint octants that are called *nodes* in the context of data structure and *cells* in the context of the 3d space. Different memory representations are possibile for limited applications, suitable for small lightweight robots (but this analysis is out of the scope of this work). In this implementation, octree's nodes are associated with *tags* to store further information about space regions. Tipically, octrees are statically built from an a-priori set of

points or object. In our application is it possible to insert and delete points and trajectories at run-time, together with tags informations, after a sensing from on-board sensors or a communication hand-skake from a neighbor robots. A set of OSP primitives are used to store, delete or search from the Octree Space Partition: `insertIntoOctree`,`deleteFromOctree`,`OctreeSearch`.

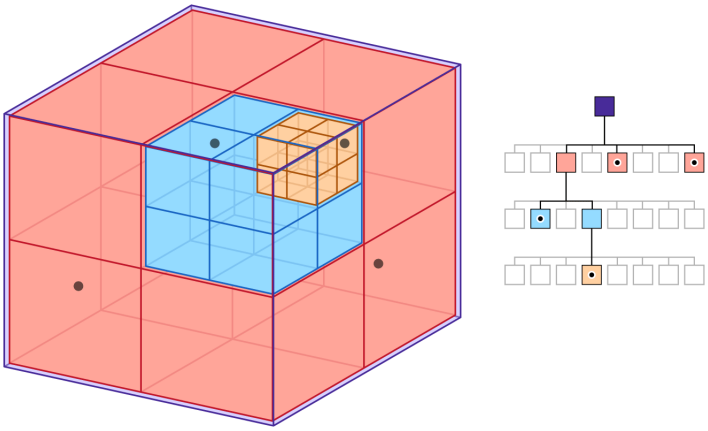


FIGURE 6.1: Each *voxel* of the OSP is equipped with a tag to store further information about space (e.g. yellow voxels are obstacles). The tags are used in the validity checking phase of the sampling-based algorithm to bias the random search. An Octree is a lightweight data structure and is suitable for low-power robots.

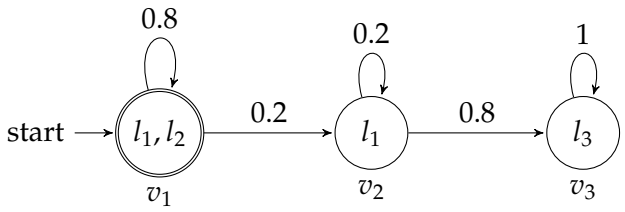
Problem Definition Given a set of quadrotor R moving in the workspace W_{free}, an Octree Space Partition for each robot, a *prior* map of obstacles stored in every Octrees, and a *local task* ϕ_i for each robot, plan a set of non collidable paths $s_i : [0, 1]$ with $i = (1 \dots N)$, such that every path satisfies ϕ_i. We assume that the agents can always communicate among them (statically connected, with fixed network topology) and that the trajectories are computed following a round-robin scheduling based on ID.

6.3 Planning Framework

To face the defined problem, we took inspiration from the synergistic framework [9][29][22] and we implement a framework composed of three layers. Some approaches use to decompose a global task, specified in LTL, for a set of robots [14]. Conversely, we use a fully distributed approach defining a local task for each robot

on a common set of LOI and we introduce a set of communication primitives. The high-level planner computes the Buchi Automata (BA) for each ϕ_i, computes the product automaton [29] between the FTS representing the actual knowledge map and the BA, then pass a plan to the low-level planner. The HL planner is also able to update the and recompute the plan after a communication update. The octree layer stores a lightweight representation of the obstacles map and the *occupancy* of the trajectory computed by different agents. The low-level sampling-based planner, finally computes feasible trajectories in SE(3) using the information provided by the octree-layer.

Abstraction Literatures about LTL planning for mobile robots challenge the problem of state explosion using different methods [43]. We use a minimalistic abstraction of the environment based on a suitable set of LOI and we shift the complexity of searching a path to the low-layer planner that use a sampling-based approach. We define a set of LOI $V = (v_1, v_2, \ldots, v_n)$ as a set of three-dimensional point $v_i \subset W_{free}$. The set of LOI V becomes the state set of an FTS, with the edge representing adjacency between point of interest. To every state is associated a set of labels from $L \cup O$, and to every edge a cost.



Every local task specification ϕ_i, that consists in an LTL formula on the set of AP (label) is converted in a Buchi Automata A^{ϕ} and the product automaton with the FTS is computed. After that, the *optimal accepting run*, as a sequence of LOI is computed on the product automaton using Dijkstra is computed and passed to the sampling-based planner.

The high level planner, based on the work of Guo [26], is sketched in Algorithm 1. The algorithm takes as input the task specification in LTL language and the Finite Transition Systems that models the environment. In line 2 it computes the Buchi Automata for the specification; in line 3 it returns a plan computing the product

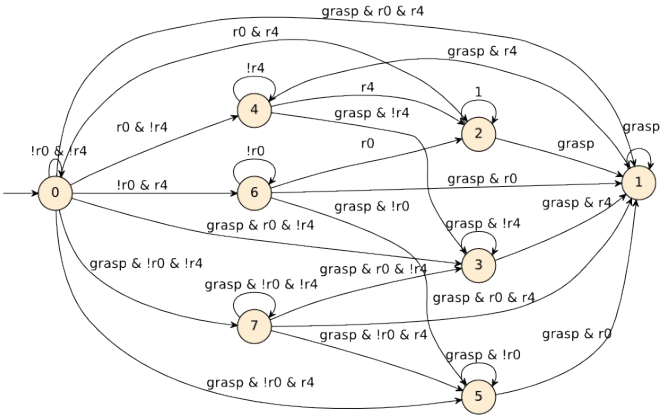


FIGURE 6.2: *Buchi Automata*: Each local task specification ϕ_i, expressed in LTL language, is converted in a buchi automata accepting FTS runs that satisfie ϕ_i. In figure the Buchi Automata for the specification $\phi_1 = \Diamond r_4 \wedge \Diamond\Box grasp \wedge \Diamond r_0$

automaton; in line 4 it starts the `HL-Comm` procedure that eventually send or receive new knowledge from neighbors agents; in line 5-7 the planner eventually update the FTS and re-compute a new plan.

Algorithm 2 High Level LTL planner

1: **procedure** LTLPLAN(ϕ_i, T_i)
2: $A^{\phi} = ComputeBuchi(\phi_i)$
3: $\tau = ComputePlan(A^{\phi}, T_i)$
4: $flag, new_k = HLComm()$
5: **if** $flag == True$ **then**
6: $T_i = UpdateFTS(T_i, new_k)$
7: $\tau = ComputePlan(A^{\phi}, T_i)$
8: **end if**
9: **end procedure**

RRT search with Octree The low-level layer use a sampling-based algorithm to compute feasible trajectories between LOI in SE(3). A sampling-based algorithm (e.g. RRT) is a data structure that randomly search for new states and connect them to the tree if they belong to W_{free} or satisfy a custom condition. We propose an octree-driven partition based approach that checks for new states against the Octree Space Partition. We built our approach mainly on the BiTRRT algorithm (Bi-directional Transition-based Rapidly-exploring Random Trees) [35], even if is possibile to use other sample-based planning algorithm. This planner grows two T-RRTs, one from the start and one from the goal, and attempts to connect the trees somewhere in the middle (RRT).

The *state validation* routine `OctreeCheck` checks if the sampled state belong to an obstacle voxel and retrieve the associated tag. The `ComputeTrajectory` routine obtain a *plan fragment* from the high level planner and compute a sample-based feasible trajectory checking states with OctreeCheck.

Algorithm 3 sketchs the Low level planner. The `ComputeTrajectory` routine takes as input the next waypoint from plan τ_i; in line 2 it gets the actual position τ_{start} of the robot from the localization systems (simulation or motion capture); in line 3 it runs the sampling-based algorithm (e.g. BiTRRT) and compute a feasible trajectory between τ_{start} and τ_i using `OctreeCheck` validator skectched in Algorithm 2. The planner has a time deadline specified in T_D. In line 4 the planner send-and-recieve computed trajectory with neighbors agents, as a set of point of SE(3), and store it in the local Octree Space Partition data structure.

Algorithm 3 Checks sampled state against Octree Space Partition

1: **procedure** OCTREECHECK(x_{new})
2: $tag = OctreeSearch(x_{new})$ ▷ if x_{new} belong to an Octree voxel retrieves the tag
3: **if** label == *obs* **then**
4: Return *True*
5: **else**
6: Return *False*
7: **end if**
8: **end procedure**

Algorithm 4 Compute SE(3) trajectories and communicate with neighbors agents

1: **procedure** COMPUTETRAJECTORY(τ_i)
2: $\tau_{start} = GetPosition()$ ▷ Retrieves the actual position from the localization system
3: $s(t) = BiTRRT(\tau_{start}, \tau_i, T_D, OctreeCheck)$
4: $LLComm(ID, s(t))$ ▷ Send $s(t)$ to the ID's Octree
5: **end procedure**

Failure and Completeness The failure of the planning can occur due three main reasons. The first is that the High Level Planner can't find a path satisfying the LTL specification. This can also occur after a knowledge update. The second reason is that there is no feasible solution due to the amount of obstacles voxels contained in the Octree. The last failure condition concerns the planning time, as every sampling-based planner. Indeed the proposed framework inherits the *probabilistical*

completeness of the RRT planning algorithm. This means that, for the problem defined in section 2, if exists an High Level plan that satisfies the specification, and if the agents can properly communicate, then the probability of failing to find the path approaches to zero as planning time increases.

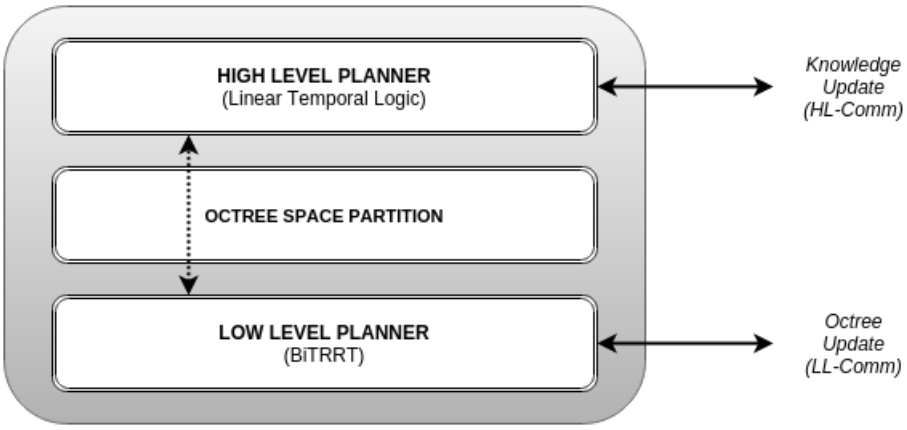


FIGURE 6.3: *Framework architecture*: The High Level layer, implementing the LTL planner, send to the Low Level layer a sequence of waypoints and wait for confirmation (position reached). The BiTRRT planner in the lower layer computes feasible trajectories validating new states against the Octree Space Partition. Communication with other agents is possible by `HL-Comm`, `LL-Comm`

6.4 Case Study

The framework is written entirely in Python. The High Level planner use the LTL library of Guo [26]. The low level planner uses OMPL (Open Motion Planning Library) for the BiTRRT algorithm and state validation. The environment and the Octree Space Partition is simulated in V-Rep. The framework use the Robot Operating System as backbone and is composed by different node running on an Ubuntu Linux machine with a QuadCore i3 processor (2.1Ghz) and 4GB of RAM. We used an OptiTrack motion capture system to execute the computed trajectory with two 10cm micro-quadrotor (Crazyflie). The robots are controlled by closing the position loop with a PID controller. The controllers are written in ROS. To test the framework we create a simulated scenario with six location of interest, two robots and two starting point. Ten obstacles was manually inserted in the Octrees. The total volume of the environment corresponds to the volume of our motion capture arena (3x3x3 meters).

Every LOI is labeled with *standard labels* $(start, r, r_0, r_1, r_2, r_3, r_4)$ and *additional labels* (`object`,`view`) related respectively to some *action proposition* like (`grasp`,`observe`). The points (r_2, r_3) are labeled rispectively with the additional labels $(view, object)$. A task specification can be formed using logical and LTL operators specified in section 6.1. In our test we set, for the two agents, the specifications

$$\phi_1 = \Diamond r_4 \wedge \Diamond\Box grasp \wedge \Diamond r_0 \tag{6.1}$$

$$\phi_2 = \Diamond r_2 \wedge \Diamond\Box observe \wedge \Diamond r_1 \tag{6.2}$$

The meaning of the specification ϕ_1 is *"eventually visit r_4, and infinitely often grasp an object, and eventually visit r_0"* while the meaning of the specification ϕ_2 is *"eventually visit r_2, and infinitely often observe a view point, and eventually visit r_1"*

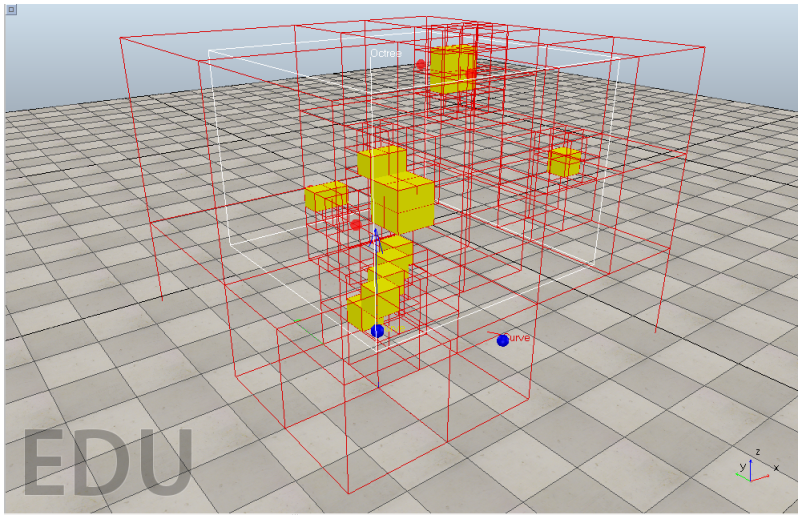


FIGURE 6.4: The Octree Space Partition stores information about the workspace. The obstacles are in *yellow*. The red sphere are labeled *location of interest* while blue sphere are the actual position of the robots. The simulated environment is built in V-Rep.

Figure 6.4 shows the a-priori map stored in the OSP while Figure 6.5 shows the computed path for the task specified in equations (6.1) and (6.2). The high level plan is computed in 3 milliseconds while the low level plan is computed in an average time of 1 millisecond even if is more computational demanding. This is due to the efficiency of OMPL that is compiled in C++. The path execution of the quadrotors takes 40 seconds in average. We manually simulate a *knowledge update* adding labels to a location of interest, by calling the `HL-Comm` routine. The framework was able to

update the FTS, replan the High Level plan and compute a new trajectory.

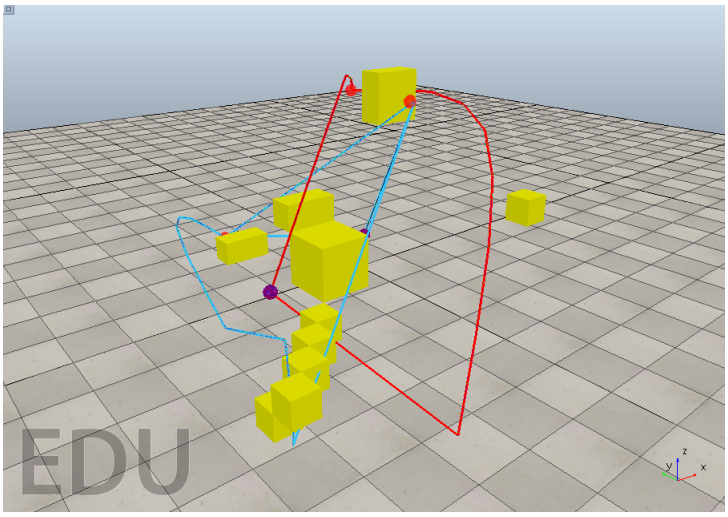


FIGURE 6.5: *Two agents simulation*: The computed paths are in *blue* and *red*, for specifications $\phi_1 = \diamond r_4 \wedge \diamond\Box grasp \wedge \diamond r_0$ and $\phi_2 = \diamond r_2 \wedge \diamond\Box observe \wedge \diamond r_1$

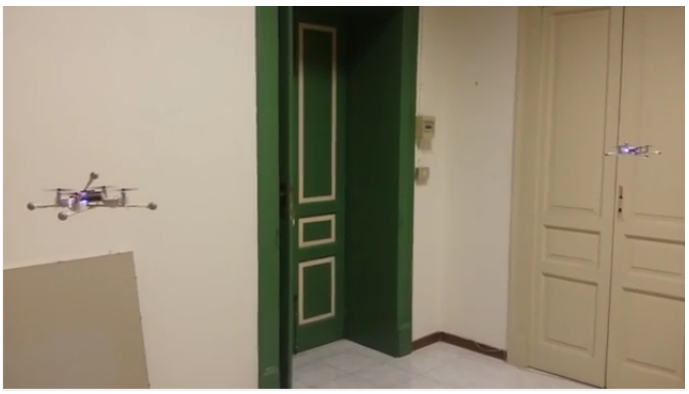


FIGURE 6.6: *Experimentation*: the V-Rep simulator is integrated in ROS (Robot Operating Systems) and drive two CrazyFlie 10cm quadrotor inside an OptiTrack motion capture system. The position control is implemented using a PID.

6.5 Conclusions and Future Work

In this chapter we presented a framework for distributed multi-agent planning from LTL specification designed for micro-quadrotors. The framework use an high level LTL planner in conjuction with a low-level sampling-based planner. The workspace is modeled as Finite Transition Systems while the sampling-based planner use an Octree Space Partition to store information about obstacles and neighbors agents.

TABLE 6.1: Computation time

Spec.	States	HL Comp. Time	LL Comp. Time	T_e *	T_D **
ϕ_1	49	3ms	1	41secs	200ms
ϕ_2	49	3ms	0.9	35secs	200ms

* Trajectory execution time inside the motion capture system
** Sampling-based planning deadline (in milliseconds)

Future work could include experiments with more agents and real moving obstacle tracked by the motion capture, realistic knowledge updates from simulated sensors or sense-and-replan behaviors, different network topologies and implementation of the planners on a set of micro-quadrotor equipped with low-power embedded linux computer and wireless communication.

Bibliography

[1] Baker A. *Matrix Groups: An Introduction to Lie Group Theory*.

[2] E. Altug, J. Ostrowski, and C.J. Taylor. "Control of a Quadrotor Helicopter Using Dual Camera Visual Feedback". In: *The International Journal of Robotics Research* 24.5 ().

[3] Minguez J. Antich J Ortiz A. "A Bug-Inspired Algorithm for Efficient Anytime Path Planning". In: *IEEE International Conference on Intelligent Robots and Systems, 2009* ().

[4] S.B. Belta C. Ayala A.I.M Andersson. "Temporal Logic Motion Planning in Unknown Environments". In: *IROS* (2013).

[5] José Raul Azinheira and Patrick Rives. "Image-based visual servoing for vanishing features and ground lines tracking: application to a UAV automatic landing". In: *International Journal of Optomechatronics* 2.3 (2008), pp. 275–295.

[6] Lin M. Manocha D. Van den Berg J. "Reciprocal Velocity Obstacle for Real-Time Multi-Agent Navigation". In: *ICRA* (2008).

[7] Kavraki L.E. Ivardi M.Y. Bhatia A. "Sampling-based Motion Planning with Temporal Goals". In: *ICRA 2010* ().

[8] Kavraki L.E. Ivardi M.Y.Motion Planning with Hybrid Dynamics Bhatia A. and Temporal Goals. In: *IEEE Conference on Decision and Control 2010* ().

[9] Maly M.R. Kavraki L.E. Ivardi M.Y. Bhatia A. "A Multi-layered Synergistic Approach to Motion Planning with Complex Goals". In: *Robotics and Automation Magazine, IEEE, vol.18, n.3* (2011).

[10] Marsden J.E. Bou-Rabee N. In: ().

[11] F. Chaumette and S. Hutchinson. "Visual Servo Control, Part 1: Basic Approaches". In: *IEEE Robotics and Automation Magazines* ().

[12] F. Chaumette and S. Hutchinson. "Visual Servo Control, Part 2: Advanced Approaches". In: *IEEE Robotics and Automation Magazines* ().

[13] F. Chaumette, P. Rives, and B. Espiau. "The Task Function Approach Applied to Vision-based Control". In: *IEEE Control Systems Magazine* ().

[14] X.C. Belta C. Chen Y. Ding. "Synthesis of Distributed Control and Communication Schemes from Global LTL Specifications". In: *CDC* (2011).

[15] A. Cherubini, F. Chaumette, and G. Oriolo. "An image-based visual servoing scheme for Following paths with nonholonomic mobile robots". In: *Control, Automation, Robotics and Vision, 2008 International Conference on*. 2008.

[16] Hutchinson S. Kantor G. Burgard W. Kavraki L. Thrun S. Choset H. Lynch K. "Principles of Robot Motion". In: *Cambridge, MA: MIT Press 2005* ().

[17] Eberly D. *Interpolation of Rigid Motions in 3D*.

[18] Eade E. *Lie Groups for 2D and 3D transformations*.

[19] B. Espiau, F. Chaumette, and P. Rives. "A New Approach to Visual Servoing in Robotics". In: *IEEE Transactions on Robotics and Automation* 8.3 (1992).

[20] Kress-Gazit H. Pappas G.J. Fainekos G.E. "Hybrid Controllers for Path Planning: A Temporal Logic Approach". In: *CDC* (2005).

[21] A. Faragasso et al. "Vision-Based Corridor Navigation for Humanoid Robots". In: *International Conference on Robotics and Automation*. 2013.

[22] Dimarogonas-D.V. Kyriakopoulos K.J. Filippidis I. "Decentralized Multi-Agent Control from Local LTL Specifications". In: *CDC 2012* ().

[23] Mallot H. Franz M. "Biomimetic robot navigation". In: *Robotics and Autonomous Systems 30, 2000* ().

[24] Galindo C. Ortiz-de-Galisteo A. Fernandez-Madrigal J.A. Moreno F.A. Gonzalez J. Blanco J.L. and Martinez J.L. "Mobile Robot Localization based on Ultra-Wide-Band Ranging: A Particle Filter Approach". In: ().

[25] N. Guenard, T. Hamel, and R. Mahony. "A Practical Visual Servo Control for an Unmanned Aerial Vehicle". In: *IEEE Transaction on Robotics* 24.2 (2008).

[26] Dimarogonas D.V. Guo M. "Distributed Plan Reconfiguration via Knowledge Transfer in Multi-Agent Systems under Local LTL Specifications". In: *ICRA 2014* ().

[27] Johansson K.H. Dimarogonas D.V. Guo M. "Motion and Action Planning under LTL Specifications using Navigation Functions adn Action Description Language". In: *IROS* (2013).

[28] Wanner G. Hairer E. Lubich C. *Geometric Numerical Integration*.

[29] Lahijanian M. Kavraki L.E. Ivardi-M.Y. He K. "Towards Manipulation Planning with Temporal Logic Specifications". In: ().

[30] Dominik Honegger et al. "An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications". In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 1736–1741.

[31] M. Hua et al. "A Control Approach for Thrust-Propelled Underactuated Vehicles and its Application to VTOL Drones". In: *IEEE Transaction on Automatic Control* 54.8 (2009).

[32] M. Hua et al. "Introduction to Feedback Control of Underactuated VTOL Vehicles". In: *IEEE Control Systems Magazine* (2013).

[33] S. Hutchinson, G.D. Hager, and P. Corke. "A Tutorial on Visual Servo Control". In: *IEEE Transactions on Robotics and Automation* 12.5 (1996).

[34] M. Hwangbo. "Vision-Based Navigation for a Small Fixed-Wing Airplane in Urban Environment". PhD thesis. Carnegie Mellon University, 2012.

[35] Cortes J. Simeon T. Jaillet L. "Sampling-Based Path Planning on Configuration-Space Costmaps". In: *IEEE Transaction of Robotics, vol.26, n.4* ().

[36] Ober-Blobaum S. Junge O. Marsden J.E. "Discrete Mechanics and Optimal Control". In: ().

[37] Rivilin E. Kamon I. Rimon E. "Range-Sensor-Based Navigation in Three-Dimensional Polyhedral Environments". In: *The International Journal of Robotics Research, January 2001* ().

[38] Rivilin E. Kamon I. Rimon E. "TangentBug: A Range-Sensor-Based Navigation Algorithm". In: *IJRR 1998* ().

[39] Frazzoli E. Karaman S. "Sampling Based Motion Planning with Deterministic mu-Calculus Specifications". In: *CDC* (2009).

[40] Belta C. Kloetzer M. "A Fully Automated Framework for Control of Linear Systems from Temporal Logic Specifications". In: *Transactions on Automatic Control, IEEE, vol.53, n.1* ().

[41] Marsden J.E. Kobilarov M. "Discrete Geometric Optimal Control on Lie Groups". In: ().

[42] Marsden J.E. Sukhtame G.V. Kobilarov M Desbrun M. "A discrete geometric optimal control framework for systems with symmetries". In: ().

[43] Wongpiromsarn T. Topcu U. Kress-Gazit H. "Correct Reactive High-Level Robot Control". In: *Robotics and Automation Magazine, IEEE* (September 2011).

[44] Dyer C. Kutulakos K. Lumelsky V. "Vision-Guided Exploration: A Step Toward General Motion Planning in Three Dimensions". In: *Int. Conf. on Robotics and Automation 1993* ().

[45] Lumelsky V. Kutulakos K. Dyer C. "Provable Strategies for Vision-Guided Exploration in Three Dimensions". In: *IEEE International Conference on Robotics and Automation, 1994* ().

[46] West M. Lall S. "Discrete variational Hamiltonian mechanics". In: *Mathematical and general 39* ().

[47] T. Lee, M. Leok, and N. Harris McClamroch. "Control of Complex Maneuvers for a Quadrotor UAV using Geometric Methods on SE(3)". In: *arXiv:1003.2005v4* ().

[48] Harris McClamroch N. Lee T. Leok M. "Lie group variational integrators for the full body problem". In: *Computeer methods in applied mechanics and engineering* ().

[49] Leok M. Lee T. Harris McClamroch N. "A Lie Group Variational Integrator for the Attitude Dynamics of a Rigid Body with applications to the 3d pendulum". In: *IEEE Conference on Control Applications 2005* ().

[50] Shingel T. Leok M. "General Techniques for constructing Variational Integrators". In: *Frontiers of Mathemaics in China* ().

[51] Marsden J.E. Ortiz M. Leyndecker S. Ober-Blobaum S. "Discrete Mechanics and Optimal Control for Constrained Systems". In: ().

[52] Stepanov A. Lumelsky V. "Path-Planning Strategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape". In: *Algorithmica 1987* ().

[53] Tiwari S. Lumelsky V. "An Algorithm for Maze Searching with Azimuth Input". In: *IEEE 1994* ().

[54] Kobilarov M. "Discrete Geometric Optimal Control of Multi-body Systems". In: ().

[55] Yi Ma. *An invitation to 3-d vision: from images to geometric models*. Vol. 26. springer, 2004.

[56] R. Mahony and T. Hamel. "Visual servoing using linear features for underactuated rigid body dynamics". In: *International Conference on Intelligent Robots and Systems*. 2001.

[57] Ratiu T. Marsden J.E. *Introduction to Mechanics and Symmetry*.

[58] Zaccaria R. Mastrogiovanni F. Sgorbissa A. "Robust Navigation in an Unknown Environment With Minimal Sensing and Representation". In: *IEEE Transactions on Systems, Man and Cybernetics, vol. 39, Feb 2009* ().

[59] D. Mellinger and V. Kumar. "Minimum Snap Trajectory Generation and Control for Quadrotors". In: *International Conference on Robotics and Automation*. 2011.

[60] Mohta K. Mulgaonkar Y. Kumar V. Nagatani K. Okada Y. Kiribayashi S. Otake K. Yoshida K. Ohno K. Takeuchi E. Michael N. Shen S. and Tadokoro S. "Collaborative mapping of an earthquake-damaged building via ground and aerial robots". In: *J. Field Robotics, vol. 29, no. 5, pp. 832–841, 2012* ().

[61] Marcos Nieto and Luis Salgado. "Real-time robust estimation of vanishing points through nonlinear optimization". In: *SPIE Photonics Europe*. International Society for Optics and Photonics. 2010, pp. 772402–772402.

[62] R. Ozawa and F. Chaumette. "Dynamic Visual Servoing with Image Moments for a Quadrotor Using a Virtual Spring Approach". In: *International Conference on Robotics and Automation*. 2011.

[63] U. Murray R.M. Wongpiromsarn T. Ozay N. Topcu. "Distributed Syntesis of Control Protocols for Smart Camera Networks". In: *IEEE/ACM Internation Conference on Cyber-Physical Systems* (2011).

[64] E. Pall et al. "Railway track following with the AR.Drone using vanishing point detection". In: *Automation, Quality and Testing, Robotics, 2014 IEEE International Conference on*. 2014, pp. 1–6.

[65] F. Pasteau, M. Babel, and R. Sekkal. "Corridor following wheelchair by visual servoing". In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE. 2013, pp. 590–595.

[66] Kavraki L.E. Ivardi M.Y. Plaku E. "Falsification of LTL Safety Properties in Hybrid Systems". In: *TACAS* (2009).

[67] H. de Plinval et al. "Visual Servoing for Underactuated VTOL UAVs: a Linear, Homography-Based Framework". In: *International Conference on Robotics and Automation*. 2011.

[68] Marino R. "Design, Modeling and Control of a Multicopter Flying Robot". In: *M.Sc. Thesis, Ecole Centrale de Nantes, Sept 2012* ().

[69] V. Raman. "Reactive Switching Protocols for Multi-Robot High-Level Tasks". In: *IROS* (2014).

[70] A. Ramirez et al. "Stability analysis of a vision-based UAV controller for autonomous road following missions". In: *2013 Internation Conference on Unmanned Aircraft Systems (ICUAS)*.

[71] Beard R.W. Atkins E.M. Ren W. In: ().

[72] LaValle S. *Planning Algorithms*.

[73] N. Michael S. Shen Y. Mulgaonkar and V. Kumar. "Multi-Sensor Fusion for Robust Autonomous Flight in Indoor and Outdoor Environments with a Rotorcraft MAV". In: *Proc. of the IEEE Intl. Conf. on Robot. and Autom., 2014* ().

[74] Ramaithitima R. Kumar V. Pappas G.J. Seshia S.A. Saha I. "Automated Composition of Motion Primitives for Multi-Robot Systems from Safe LTL Specifications". In: *IROS* (2014).

[75] H. Samet. "An overview of Quadtrees, Octrees and related hierarchical data structures". In: (1988).

[76] Shaojie Shen, Nathan Michael, and Vijay Kumar. "Autonomous multi-floor indoor navigation with a computationally constrained MAV". In: *Robotics and automation (ICRA), 2011 IEEE international conference on*. IEEE. 2011, pp. 20–25.

[77] Kumar V. Shen S. Michael N. "Autonomous Multi-floor Indoor Navigation with a Computational Constrained MAV". In: *ICRA 2011* ().

[78] Laubach S.L. "Theory and Experiments in Autonomous Sensor-Based Motion Planning with Application for Flight Planetary Microrovers". In: *Ph.D. Thesis, Caltech 1999* ().

[79] Tumova J. Belta C. Rus D. Smith S.L. "Optimal Path Planning for Surveillance with Temporal Logic Contraints". In: *IJRR* (2011).

[80] Li C. Tang K. and Chiu S. "An Electronic-Nose Sensor Node Based on a Polymer-Coated Surface Acoustic Wave Array for Wireless Sensor Network Applications". In: *Sensors 2011* ().

[81] LaValle S. Taylor K. "I-Bug: An intensity-based bug algorithm". In: *Proceedings IEEE International Conference on Robotics and Automation, 2009* ().

[82] C. Teulier et al. "3D model-based tracking for UAV position control". In: *International Conference on Intelligent Robots and Systems*. 2010.

[83] J. Thomas et al. "Toward Image Based Visual Servoing for Aerial Grasping and Perching". In: *International Conference on Robotics and Automation*. 2014.

[84] W. Thomas. *Automata and Reactive Systems*.

[85] Ober-Blobaum S. Trachtler A. Timmermann J. Khatab S. "Discrete Mechanics and Optimal Control and its Application to a Double Pendolum on a Cart". In: ().

[86] J.M Toibero et al. "Switching Visual Servoing Approach for Stable Corridor Navigation". In: *International Conference on Advanced Robotics*. 2009.

[87] Berhoz A. Meyer J. Trullier O. Wiener S. "Biologically-based artifical navigation systems: Review and prospects". In: *Prog. Neurobiol. vol.51, 1997* ().

[88] Belta C. Vasile C.I. "Reactive Sampling-Based Temporal Logic Path Planning". In: *ICRA* (2014).

[89] M. Westenkirchner. "Image-Based Controller for MAVs". MA thesis. ETH Zurich, 2010.

[90] Konstantin Yakovlev et al. "Distributed Control and Navigation System for Quadrotor UAVs in GPS-Denied Environments". In: *Proceedings of the 7th IEEE International Conference Intelligent Systems*. 2014.

[91] H. Zhang and J. Ostrowski. "Visual Servoing with Dynamics: Control of an Unmanned Blimp". In: *International Conference on Robotics and Automation*. 1999.

[92] Song J. Li X. Zhu Y. Zhang T. "A New Bug-type Navigation Algorithm Considering Practical Implementation Issues for Mobile Robots". In: *IEEE International Conference on Robotics and Biomimetics, 2010* ().